



**UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA**

**AEROPÊNDULO, PROTOTIPAGEM E SIMULADOR GRÁFICO COMO
FERRAMENTA PARA ESTUDOS DE TÉCNICAS DE CONTROLE E MODELAGEM
POR IDENTIFICAÇÃO DE SISTEMAS**

OSÉIAS DIAS DE FARIAS

TUCURUÍ-PA

2023

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

OSÉIAS DIAS DE FARIAS

AEROPÊNDULO, PROTOTIPAGEM E SIMULADOR GRÁFICO COMO FERRAMENTA
PARA ESTUDOS DE TÉCNICAS DE CONTROLE E MODELAGEM POR
IDENTIFICAÇÃO DE SISTEMAS

Trabalho de conclusão de curso apresentado
ao colegiado da Faculdade de Engenharia Elétrica,
do Campus Universitário de Tucuruí, da
Universidade Federal do Pará, como requisito
necessário para obtenção do título de Bacharel
em Engenharia Elétrica.

Orientador: Prof. Dr. Raphael Barros Teixeira

TUCURUÍ-PA

2023

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

**AEROPÊNDULO, PROTOTIPAGEM E SIMULADOR GRÁFICO COMO
FERRAMENTA PARA ESTUDOS DE TÉCNICAS DE CONTROLE E MODELAGEM
POR IDENTIFICAÇÃO DE SISTEMAS**

AUTOR: OSÉIAS DIAS DE FARIAS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À BANCA EXAMINADORA APROVADA PELO COLEGIADO DA FACULDADE DE ENGENHARIA ELÉTRICA, SENDO JULGADO

.....

BANCA EXAMINADORA:

Prof. Dr. Raphael Barros Teixeira
Orientador / UFPA-CAMTUC-FEE

Prof. Dr. NOME PRIMEIRO AVALIADOR
Membro 1 / UFPA-CAMTUC-FEE

Prof. Dr. NOME PRIMEIRO AVALIADOR
Membro 2 / UFPA-CAMTUC-FEE

Resumo

Este trabalho apresenta um ecossistema abrangente para o estudo de sistemas de controle, que combina a integração de um protótipo físico, um simulador 3D e uma interface gráfica interativa. O objetivo principal é oferecer uma plataforma que permita investigar várias facetas dos sistemas de controle, incluindo identificação de sistemas, desenvolvimento de controladores, otimização de controladores, aplicação de Inteligência Artificial, etc. A motivação para este projeto reside na intrínseca complexidade associada à compreensão de sistemas de controle, que frequentemente apresenta desafios, sobretudo para estudantes que precisam transpor a barreira de abstrair sistemas físicos em termos de equações matemáticas. No entanto, é importante destacar que a aplicabilidade deste projeto transcende a mera utilização por iniciantes, uma vez que se revela uma ferramenta versátil e robusta também para pesquisas mais avançadas e complexas na área. A combinação de protótipos, simuladores, interface gráfica e integração de diversas disciplinas da Engenharia Elétrica cria um ambiente de aprendizado envolvente e eficaz para o estudo de sistemas de controle. Essa abordagem reduz a resistência inicial dos alunos e promove uma compreensão mais profunda dos conceitos e aplicações dessa área. Por meio da prática, os estudantes podem perceber a relevância das abstrações matemáticas na resolução de problemas reais, preparando-os para lidar com sistemas de controle complexos e desafiadores.

Palavras Chave: Aeropêndulo, identificação de sistema, protótipo, simulador, programação.

Abstract

Texto do abstract (inglês)

Keywords: Palavras chave em inglês.

SUMÁRIO

Resumo	iv	
Abstract	v	
Sumário	vi	
1	INTRODUÇÃO	1
1.1	Justificativa	1
1.2	Objetivos	2
1.2.1	Objetivos Gerais	2
1.2.2	Objetivos Específicos	2
1.3	Escopo do Trabalho	3
2	DESENVOLVIMENTO	4
2.1	Fundamentação Teórica	4
2.1.1	Modelo Matemático do Motor CC Série	4
2.1.1.1	Modelagem da Parte Mecânica do Motor CC Série	6
2.1.1.2	Modelagem da Parte Elétrica do Motor CC Série	8
2.1.1.3	Linearização do modelo do Motor CC Série	9
2.1.2	Modelo Matemático Braço do Aeropêndulo	11
2.1.3	Junção dos subsistemas	14
2.2	Implementação do Protótipo	15
2.2.1	Protótipo Aeropêndulo	15
2.2.2	Parte estrutural do sistema	15
2.2.3	Parte Elétrica do sistema	16
2.2.3.1	Potenciômetro $50k\Omega$	17
2.2.3.2	Placa de desenvolvimento Esp32	17
2.2.3.3	Fonte Chaveada 2A, 5V, 25W	18
2.2.3.4	Módulo Driver L298n	18
2.2.3.5	Conjunto (suporte motor cw/ccw hélice) para drones fpv racing quadcopter	19
2.2.3.6	Componentes eletrônicos (resistivos e capacitivos)	20
2.2.4	Montagem do Protótipo	20
2.2.4.1	Parte Física	20

2.2.4.2	Parte Elétrica	20
2.3	Desenvolvimento dos Softwares	22
2.3.1	Linguagens Python, C e C++	22
2.3.2	Gêmeo Digital	23
2.3.2.1	Biblioteca VPython	23
2.3.2.2	Arquitetura do Gêmeo Digital	24
2.3.3	Interface Gráfica para Configuração, visualização e aquisição de dados da Planta	24
2.3.3.1	Biblioteca CustomTkinter e Matplotlib	25
2.3.3.2	Biblioteca PySerial, NumPy e Pandas	26
2.3.3.3	Descrição das partes da interface gráfica	26
2.3.4	Firmware do microcontrolador	28
2.3.4.1	Biblioteca ler_escrever_serial	29
2.3.4.2	Biblioteca referencia	29
2.3.4.3	Biblioteca conversor	29
2.3.4.4	Biblioteca controlador_pid	29
2.3.4.5	Arquivo Principal mian.cpp	30
2.4	Ecossistema do Projeto	30
3	RESULTADOS E DISCUSSÕES	31
3.1	Desenvolvimento do Protótipo e Softwares	31
3.2	Identificação de sistema aplicado ao Aeropêndulo	31
3.2.0.1	Aplicação do Sinal PRBS na Entrada do Sistema	31
3.2.0.2	Sinal de Saída Gerado a Partir do Sinal PRBS de Entrada	32
3.2.1	Identificação Usando Mínimos Quadrados	32
3.2.1.1	Passos para realização da identificação	33
3.2.1.2	Identificação Usando Função de Transferência Discreta de Segunda, Quinta e Décima Ordem	33
3.3	Ensaio em Malha Fechada com Controlador PID	39
3.3.1	Onda Quadrada	39
3.3.2	Onda Dente de Serra	41
4	CONCLUSÃO	42
4.1	Considerações Finais	42
4.2	Trabalhos Futuros	42
REFERÊNCIAS		43



INTRODUÇÃO

1.1 Justificativa

Sistemas de controle têm como finalidade modelar, analisar e projetar controladores para que um sistema possa atender a requisitos de projeto específicos. Para atingir esse objetivo, é necessário aplicar técnicas que permitam abstrair o comportamento do sistema em termos de equações matemáticas. No entanto, é importante lembrar que, ao abstrair sistemas físicos dessa maneira, o preço pago está na percepção e interpretação da dinâmica do sistema.

Além disso, a implementação de controladores requer a expertise em diferentes áreas da engenharia, tais como eletrônica analógica e digital, programação, processamento de sinais, circuitos elétricos, entre outras. Dessa forma, torna-se necessário integrar conhecimentos multidisciplinares para a implementação bem-sucedida de controladores em sistemas reais.

No entanto, no estudo de sistemas de controle, é comum que os discentes enfrentem desafios em seu primeiro contato com a área, especialmente devido à necessidade de aplicar abstrações matemáticas para representar a dinâmica de sistemas físicos. Essa etapa inicial pode parecer complexa, porém é crucial para a compreensão e domínio dos conceitos fundamentais envolvidos na análise e controle de sistemas.

Uma das principais razões pelas quais os estudantes encontram dificuldades é a transição do mundo físico para o mundo matemático abstrato, onde os sistemas são modelados por equações diferenciais, transformadas de Laplace e outros formalismos matemáticos. Para muitos, essa mudança pode parecer distante da realidade observada, o que pode causar alguma resistência inicial.

No entanto, para superar essa barreira de aprendizagem, é fundamental buscar métodos que possibilitem aos discentes visualizar a dinâmica desses sistemas de forma interessante. Uma

abordagem promissora é a utilização de protótipos, que permitem aos alunos observar a dinâmica analisada matematicamente no mundo real. Essa aplicação prática fornece uma conexão mais tangível entre os conceitos abstratos e suas aplicações concretas, tornando o aprendizado mais envolvente e compreensível.

Adicionalmente, o uso de simuladores pode ser altamente benéfico. Ao inserir as respostas obtidas por meio dos modelos matemáticos como entrada nos simuladores, o processo de aprendizado integra ferramentas matemáticas e de visualização. Dessa forma, os alunos podem interagir com os sistemas em diferentes cenários, observando como as variáveis influenciam o comportamento dos sistemas de controle. Essa abordagem interativa e experimental ajuda a solidificar conceitos e aprimorar a compreensão do funcionamento desses sistemas complexos.

Ao combinar a teoria matemática com a prática através de protótipos e simuladores, o processo de aprendizado torna-se mais fluido e estimulante. Os alunos podem perceber a relevância das abstrações matemáticas na resolução de problemas reais, o que reduzirá a resistência inicial e aumentará o interesse pela área de sistemas de controle.

1.2 Objetivos

1.2.1 Objetivos Gerais

melhore a escrita focada para um tcc "Este trabalho tem como objetivo desenvolver um ecossistema abrangente para o estudo de sistemas de controle. Para alcançar esse propósito, será criado um projeto que integra um protótipo, um simulador e uma interface gráfica, possibilitando sua utilização na investigação de diversos aspectos e subáreas relacionadas a sistemas de controle. Isso inclui a identificação de sistemas, o desenvolvimento de controladores, a otimização de controladores, bem como a aplicação de Inteligência Artificial em sistemas de controle, entre outros tópicos.

Além disso, nossa proposta envolve a aplicação prática dos conhecimentos adquiridos durante a graduação, unindo técnicas de sistemas de controle, eletrônica analógica e digital, programação, física e cálculo em uma configuração de sistema físico. O principal objetivo é criar um ambiente que permita ao pesquisador aplicar e aprimora seus conhecimentos e observar o comportamento da dinâmica do sistema em um ambiente real. Para realizar essa tarefa, será necessário combinar tecnologias de diversas disciplinas do curso de Engenharia Elétrica, tornando o projeto ainda mais intrigante e desafiador.

1.2.2 Objetivos Específicos

- Desenvolver o protótipo de Aeropêndulo;
- Desenvolver um simulador 3D (Gêmeo Digital);

- Desenvolver uma interface gráfica com menu para manipular o comportamento do sistema em tempo real;
- Obter um modelo da planta por identificação de sistemas;
- Projetar controlador e implementar na planta;
- Avaliação do correto funcionamento do ecossistema proposto.

1.3 Escopo do Trabalho

O projeto parte de uma modelagem matemática usando como base os princípios da física newtoniana com o intuito de demonstrar que para sistemas relativamente complexos, essa técnica de modelagem pode se tornar trabalhosa e por muitas vezes impraticáveis, a partir dessa premissa parte-se para o desenvolvimento do protótipo, o objetivo está na utilização do sistema físico para aplicar o método de identificação de sistema que consiste em obter um modelo matemático, que descreva a dinâmica do sistema físico de forma aproximada, a partir dos dados de entrada e saída do protótipo.

No processo de condução dos ensaios e na aquisição dos dados essenciais para a realização dos estudos, o projeto inclui o desenvolvimento de uma interface gráfica. Esta interface capacita o pesquisador a executar ensaios e a armazenar os dados obtidos, viabilizando a subsequente modelagem e análise do comportamento do sistema em estudo. Adicionalmente, a interface é equipada com um menu que permite a configuração em tempo real dos ensaios para diferentes sinais de entrada, além da configuração dos parâmetros de frequência, amplitude e offset. Essa funcionalidade dinâmica e flexível aprimora significativamente a condução da pesquisa, tornando-a mais produtiva e eficiente.

Além da interface gráfica executada no computador, é fundamental o desenvolvimento de um firmware que gerencie aspectos cruciais, como a comunicação, aquisição de dados e controle do Aeropêndulo. Essas funcionalidades são implementadas por um microcontrolador. Este microcontrolador não apenas estabelece a comunicação com a interface gráfica através da conexão USB, mas também desempenha um papel vital na geração dos sinais de entrada para a planta e na leitura do ângulo do braço do Aeropêndulo.

Além disso, o projeto incorpora um gêmeo digital do sistema físico, na forma de um simulador 3D do Aeropêndulo. Essa abordagem permite a integração da dinâmica do sistema real com o ambiente computacional, possibilitando a recriação em tempo real da planta de forma virtual. Essa simulação em 3D não apenas enriquece a experiência de aprendizado, mas também oferece aos estudantes a oportunidade de se envolver em um processo de aprendizado prático que se assemelha significativamente às condições do mundo real. Isso resulta em um processo educacional mais imersivo e eficaz, capacitando os alunos a compreender e explorar a complexidade dos sistemas de controle de maneira mais aprofundada.



DESENVOLVIMENTO

2.1 Fundamentação Teórica

O processo que envolve modelagem de sistemas físicos em termos de equações matemáticas é uma das partes mais importantes no estudo de sistemas de controle. Segundo [Ogata \(2014, p. 11\)](#), O modelo matemático de um sistema dinâmico é definido como um conjunto de equações que representa a dinâmica do sistema com precisão ou, pelo menos, razoavelmente bem.

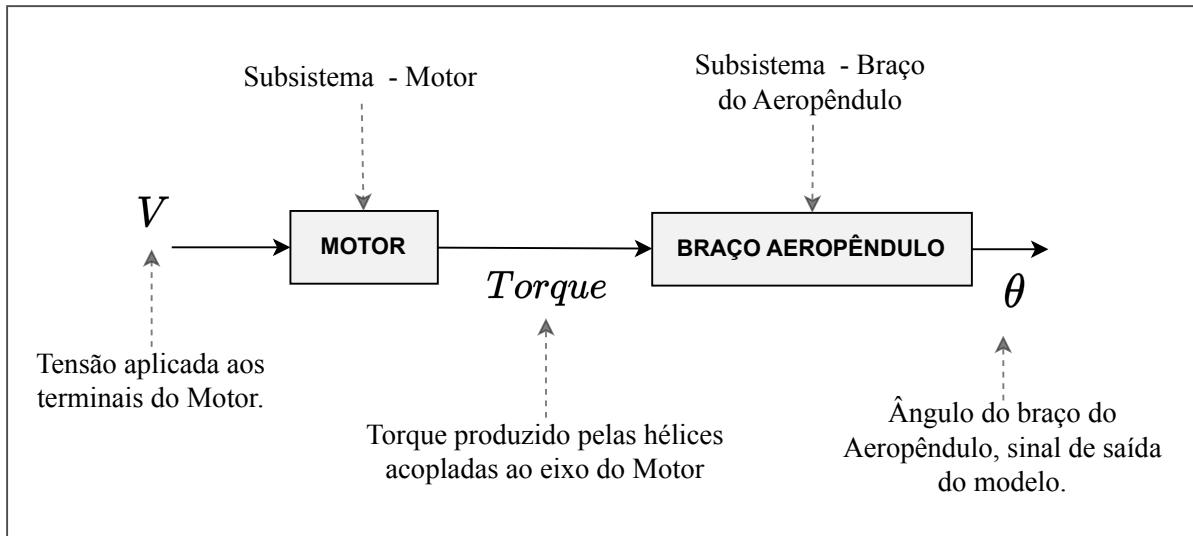
A dinâmica de muitos sistemas mecânicos, elétricos, térmicos, econômicos, biológicos ou outros pode ser descrita em termos de equações diferenciais. Essas equações diferenciais são obtidas pelas leis físicas que regem dado sistema — por exemplo, as leis de Newton para sistemas mecânicos e as leis de Kirchhoff para sistemas elétricos. Devemos sempre ter em mente que construir modelos matemáticos adequados é a parte mais importante da análise de sistemas de controle como um todo, [Ogata \(2014, p. 11\)](#).

Para realizar a modelagem do Aeropêndulo pode-se aplicar diferentes métodos, uma abordagem inicial para modelar sistemas físicos consiste em empregar as leis fundamentais da física. Além disso, em situações mais complexas, é viável decompor o sistema em subsistemas menores e, em seguida, desenvolver modelos para cada um deles. Por fim, é possível conectar esses modelos, de forma a obter uma representação aproximada do sistema real. Dessa maneira, podemos obter uma compreensão mais aprofundada e abrangente da dinâmica do sistema em questão. A Figura 1 mostra um diagrama da representação do Aeropêndulo como um conjunto de subsistemas.

2.1.1 Modelo Matemático do Motor CC Série

Os motores CC série tem como principal característica possuir o enrolamento de campo em série com o enrolamento de armadora, essa configuração resulta em um motor com torque de partida

Figura 1 – Subsistemas do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

alto, porém, o torque reduz a medida que a velocidade aumenta devido ao aumento da Força Eletromotriz FEM. Por conta desse aumento de FEM os motores CC Séries tem uma regulação de velocidade ruim, quando se aumenta a carga no eixo do motor a velocidade é reduzida que por sua vez reduz a FEM e então o torque aumenta para conseguir atuar na carga.

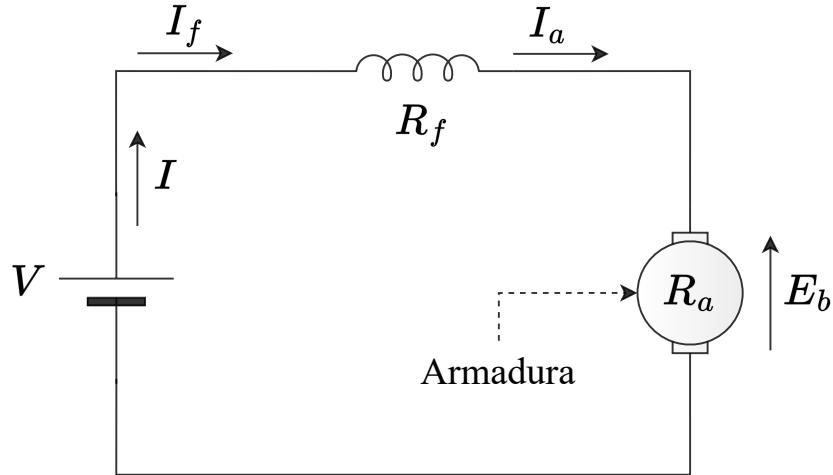
No motor série, o aumento de carga é acompanhado por elevações da corrente, da FMM de armadura e do fluxo de campo do estator (desde que o ferro não esteja completamente saturado). Como o fluxo aumenta com a carga, a velocidade deve cair para se manter o equilíbrio entre a tensão aplicada e a força contraeletromotriz. Além disso, o aumento na corrente de armadura, causado pelo aumento de conjugado, é menor do que no motor em derivação devido ao aumento de fluxo. O motor série é, portanto, um motor de velocidade variável, [Umans \(2014, p. 410\)](#).

No entanto, mesmo motores cc série com dimensões reduzidas geram torques altos com baixo consumo de corrente. Visando melhorar seu desempenho, é possível projetar controladores de malha fechada capazes de tornar esses motores mais eficientes na regulação de velocidade.

([LICEAGA-CASTRO et al., 2017](#)) foi usando como base para realizar a modelagem o motor CC série, A Figura 2 mostra um diagrama da configuração do motor CC Série, no qual o enrolamento de campo está conectado em série com o enrolamento de armadura, dessa forma, a corrente de campo é igual a corrente de armadura $i = i_f = i_a$.

Na Figura 3, mostra o diagrama eletromecânico do motor, nele pode-se observar que os componentes elétricos estão todos em série, no qual o enrolamento de campo possui uma parte resistiva e outra indutiva, assim como o enrolamento de armadura, já a parte mecânica possui

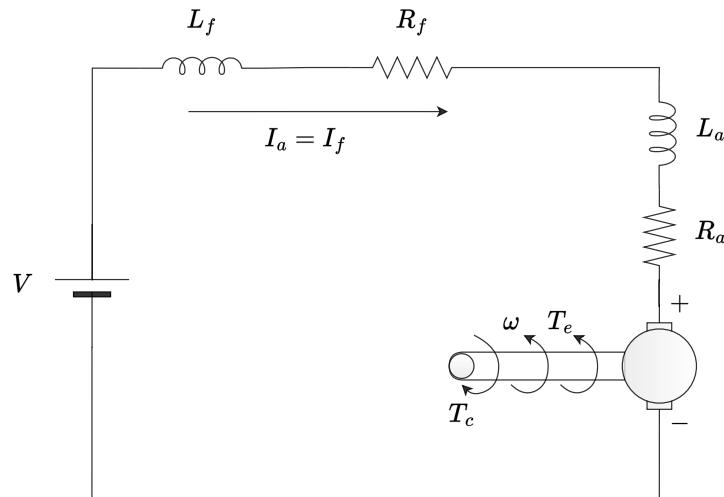
Figura 2 – Motor CC Série.



Fonte: elaborado pelo autor (2023).

uma velocidade angular dada por $\dot{\omega}$, torque eletromagnético do motor dado por T_e e torque da Carga T_c .

Figura 3 – Diagrama Elétrico/Mecânico Motor CC Série.



Fonte: elaborado pelo autor (2023).

2.1.1.1 Modelagem da Parte Mecânica do Motor CC Série

Primeiramente a parte mecânica do motor será modelada, um motor cc série é composto por uma parte rotativa "armadura", de modo que, essa parte, gera um momento de inércia J_m no eixo do motor e um fator de amortecimento viscoso b , além disso, o eixo possui uma velocidade angular $\dot{\omega}$.

Assim, a equação 2.1, obtida a partir de (LICEAGA-CASTRO *et al.*, 2017), descreve o modelo mecânico do motor CC Série.

$$J_m \ddot{\omega}(t) = T_e(t) - b\dot{\omega}(t) - T_c(t) \quad (2.1)$$

Ao expressar o torque $T_e(t)$ em função das outras variáveis, obtém-se a equação 2.2.

$$T_e(t) = J_m \ddot{\omega}(t) + b\dot{\omega}(t) + T_c(t) \quad (2.2)$$

Onde:

- T_e : Torque Eletromagnético produzido pelo Motor;
- J_m : Momento de Inércia do Eixo do Motor;
- $\ddot{\omega}$: Aceleração Angular do Eixo do Motor;
- $\dot{\omega}$: Velocidade Angular do Eixo do Motor;
- b : Fator de Amortecimento Viscoso;
- T_c : Torque de Carga.

Tanto a Força Eletromotriz $E_A(t)$ quanto o Torque Eletromagnético $T_e(t)$ dependem do fluxo magnético do entreferro Φ , com isso, tem-se as equações 2.3 e 2.4.

$$E_a(t) = \dot{\omega}(t)\Phi(i) \quad (2.3)$$

$$T_e(t) = i(t)\Phi(i) \quad (2.4)$$

O Fluxo magnético depende da corrente $i(t)$, assim, as equações 2.1 e 2.2 são não-lineares. Além disso, pode-se aproximar o fluxo Φ por uma relação linear, K_0 , ao desprezar a saturação magnética.

$$\Phi(i) = K_0 i(t) \quad (2.5)$$

A constante K_0 é a indutância mútua entre a armadura e o enrolamento de campo. Agora pode-se encontrar o modelo não-linear da parte mecânica do Motor CC Série, substituindo 2.5 em 2.4, obtém-se a expressão 2.7:

$$T_e(t) = i(t)K_0 i(t) \quad (2.6)$$

$$T_e(t) = K_0 i^2(t) \quad (2.7)$$

Substituindo 2.7 em 2.2, é possível encontrar o modelo da parte mecânica do motor CC série, assim, tem-se a expressão 2.8.

$$K_0 i^2(t) = J_m \ddot{\omega}(t) + b \dot{\omega}(t) + T_c(t) \quad (2.8)$$

2.1.1.2 Modelagem da Parte Elétrica do Motor CC Série

Para a parte elétrica, utilizou-se a lei de Kirchhoff das tensões para modelar o sistema. assim como a parte mecânica, o modelo da parte elétrica foi baseado de (LICEAGA-CASTRO *et al.*, 2017), o motor em questão é um motor de corrente contínua CC série, Dessa forma, $i_a = i_f$, portanto, obtém-se a expressão 2.9.

$$V(t) = (R_a + R_f)i(t) + (L_a + L_f)\frac{d}{dt}i(t) + E_a \quad (2.9)$$

Onde:

- V : Tensão da Fonte;
- R_a : Resistência da Armadura;
- R_f : Resistência de Campo;
- i_a : Corrente da Armadura;
- i_f : Corrente de Campo;
- E_A : Tensão Contro Eletromotriz Gerada pela Armadura;
- L_a : Impedância da Armadura;
- L_f : Impedância de Campo.

Como os componentes elétricos então em série, pode-se obter uma resistência total assim como uma indutância:

$$R = R_a + R_f \quad (2.10)$$

$$L = L_a + L_f \quad (2.11)$$

$$V(t) = Ri(t) + L\frac{d}{dt}i(t) + E_a \quad (2.12)$$

Substituindo 2.5 em 2.3, obtém-se a equação 2.13.

$$E_a(t) = \dot{\omega}(t)K_0i(t) \quad (2.13)$$

Por fim, pode-se encontrar a equação que descreve a parte elétrica do sistema ao substituir 2.13 em 2.12, assim, obtendo a equação 2.14.

$$V(t) = Ri(t) + L\frac{d}{dt}i(t) + \dot{\omega}(t)K_0i(t) \quad (2.14)$$

Dessa forma, as equações que modelam um Motor CC série são expressas por 2.15 e 2.16, em que 2.15 esta relacionada a parte mecânica e 2.16 a parte elétrica.

$$K_0i^2(t) = J_m\ddot{\omega}(t) + b\dot{\omega}(t) + T_c(t) \quad (2.15)$$

$$V(t) = Ri(t) + L\frac{d}{dt}i(t) + \dot{\omega}(t)K_0i(t) \quad (2.16)$$

2.1.1.3 Linearização do modelo do Motor CC Série

Para projetos de controle lineares é preciso linearizar o modelo encontrado, para isso, vamos reorganizar as equações 2.15 e 2.16, assim obtêm-se as equações 2.17 e 2.18:

$$\ddot{\omega}(t) = \frac{K_0}{J_m}i^2(t) - \frac{b}{J_m}\dot{\omega}(t) - \frac{1}{J_m}T_c(t) \quad (2.17)$$

$$\frac{d}{dt}i(t) = \frac{R}{L}i(t) - \frac{K_0}{L}\dot{\omega}(t)i(t) + \frac{1}{L}V(t) \quad (2.18)$$

reescrevendo as equações de estados na forma matricial,

$$x_1 = \dot{\omega} \quad (2.19)$$

$$x_2 = i \quad (2.20)$$

Coeficientes da equação de estado 2.17.

$$a_1 = \frac{K_0}{J_m}; \quad a_2 = \frac{b}{J_m}; \quad a_3 = \frac{1}{J_m} \quad (2.21)$$

Coeficientes da equação de estado 2.18.

$$b_1 = \frac{R}{L}; \quad b_2 = \frac{K_0}{L}; \quad b_3 = \frac{1}{L} \quad (2.22)$$

Representação no espaço de estados do motor CC Série não-linear

$$\dot{x}_1 = a_1 x_2^2 - a_2 x_1 - a_3 T_c \quad (2.23)$$

$$\dot{x}_2 = -b_1 x_2 - b_2 x_1 x_2 + b_3 V \quad (2.24)$$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_1 x_2^2 - a_2 x_1 - a_3 T_c \\ -b_1 x_2 - b_2 x_1 x_2 + b_3 V \end{bmatrix} = f(x, u) \quad (2.25)$$

O ponto de equilíbrio (x_1^0, x_2^0) das equações 2.23 e 2.24 é encontrado zerando as derivadas, como mostrado em 2.26 e 2.27.

$$x_2^0 = \sqrt{\frac{a_2 x_1^0 + a_3 T_c}{a_1}} \quad (2.26)$$

$$V = \frac{x_2^0(b_1 + b_2 x_1^0)}{b_3} \quad (2.27)$$

A linearização de 2.23 e 2.24 em torno do ponto de equilíbrio é obtida encontrando o jacobiano das mesmas, assim se obtêm as matrizes A e B .

$$\dot{x} = Ax + Bu \quad (2.28)$$

$$y = Cx \quad (2.29)$$

Em que,

$$A = \frac{\partial f(x, y)}{\partial x} \Big|_{x_1^0, x_2^0} = \begin{bmatrix} -a_2 & 2a_1 x_2^0 \\ -b_2 x_2^0 & -(b_1 + b_2 x_1^0) \end{bmatrix} \quad (2.30)$$

$$B = \frac{\partial f(x, y)}{\partial u} \Big|_{x_1^0, x_2^0} = \begin{bmatrix} -a_3 & 0 \\ 0 & b_3 \end{bmatrix} \quad (2.31)$$

$$C = [1, 0] \quad (2.32)$$

$$(2.33)$$

Onde,

$$u = \begin{bmatrix} T_c \\ V \end{bmatrix} \quad (2.34)$$

$$y = \dot{\omega}(t) \quad (2.35)$$

Se o torque de carga for considerado zero, $T_c = 0$, temos as seguintes expressões,

$$x_2^0 = \sqrt{\frac{a_2 x_1^0}{a_1}}; \quad V = \frac{x_2^0(b_1 + b_2 x_1^0)}{b_3}; \quad B = \begin{bmatrix} 0 \\ b_3 \end{bmatrix} \quad (2.36)$$

Agora é possível encontrar a função de transferência $G(s)$ associada as matrizes de estados A, B, C linearizadas, para isso usa-se a expressão 2.37.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = C(sI - A)^{-1}B \quad (2.37)$$

Substituindo as matrizes A, B e C em 2.37, chega-se a expressão 2.38.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \left(s \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -a_2 & 2a_1 x_2^0 \\ -b_2 x_2^0 & -(b_1 + b_2 x_1^0) \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} -a_3 & 0 \\ 0 & b_3 \end{bmatrix} \quad (2.38)$$

Assim, obtém-se a função de transferência linearizada em função dos parâmetros do motor CC Série, como mostra a equação 2.39.

$$G(s) = \frac{\dot{\omega}(s)}{V(s)} = \frac{2K_0 \sqrt{\frac{bx_1^0}{K_0}}}{J_m L s^2 + 3K_0 b x_1^0 + Rb + s (J_m K_0 x_1^0 + J_m R + Lb)} \quad (2.39)$$

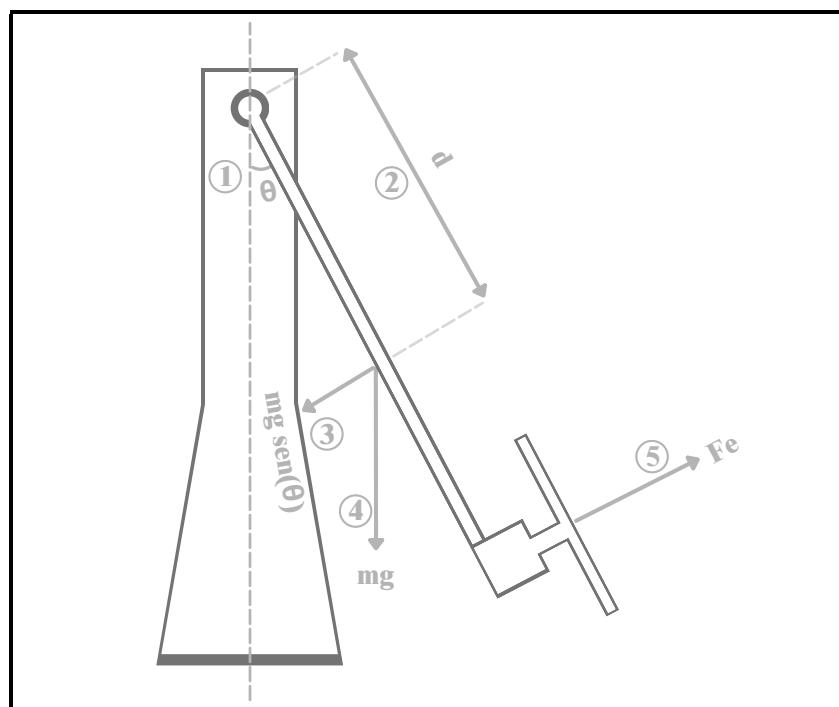
2.1.2 Modelo Matemático Braço do Aeropêndulo

Os sistemas mecânicos têm como uma de suas características os graus liberdade, da sigla em inglês 6DoF - (Six degrees of freedom) que em português quer dizer (Seis graus de liberdade), com esses seis graus de liberdade é possível mover a planta em todas as direções e ângulos, o aeropêndulo tem um grau de liberdade, sendo o ângulo θ , assim o sistema tem apenas uma variável a ser

controlada a partir do empuxo gerado pelas hélices, que por sua vez depende da velocidade angular do eixo do motor CC Série acoplado ao braço do aeropêndulo. dessa forma, podemos analisar o sistema a partir da entrada e da saída, sendo a entrada a tensão no motor e a saída o ângulo θ do braço do aeropêndulo.

como discutido antes, pode-se dividir o sistema em dois subsistemas, o motor CC Série e o braço do aeropêndulo. para o motor CC Série, já foi realizado a modelagem na sessão 2.1.1, para essa sessão será realizada a modelagem do braço do aeropêndulo, ([MOHAMMADBAGHERI; YAGHOOBI, 2011](#)) foi usando como base para se obter a equação que modela a dinâmica do braço.

Figura 4 – Diagrama esquemático do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

O modelo matemático do braço do aeropêndulo é derivado a partir das leis de Newton e do momento angular, como mostra ([MOHAMMADBAGHERI; YAGHOOBI, 2011](#)). assim, se obtém a equação 2.40.

$$F_e = J_b \ddot{\theta} + c\dot{\theta} + mgd \sin \theta \quad (2.40)$$

Onde:

- F_e : Empuxo gerado pela hélice
- J_b : Momento de inércia do Braço

- θ : posição angular do Aeropêndulo
- c : coeficiente de amortecimento viscoso
- m : massa do Aeropêndulo
- d : a distância entre o centro de massa e o ponto de pivô

A entrada do subsistema do braço do aeropêndulo é a força de empuxo proporcionada pela hélice, porém, o modelo do motor CC Série tem como saída a velocidade angular, dessa forma é preciso encontrar uma relação entre a velocidade $\dot{\omega}$ e o empuxo F_e , essa relação é não linear como mostra [xx], $F_e = K_m \dot{\omega}^2$, porém é possível aproximar por uma relação linear, $F_e = K_m \dot{\omega}$. Com isso, pode-se relacionar a velocidade angular $\dot{\omega}$ com o empuxo F_e gerado pela hélice do aeropêndulo.

O modelo encontrado tem uma parcela não linear dada por $\sin \theta$, para aplicar técnicas de projeto de controle é preciso obter o modelo linearizado da planta, isso pode ser feito considerando $\sin \theta \approx \theta$ para pequenas variações em torno de θ . dessa forma, temos a seguinte linearização:

$$F_e = J_b \ddot{\theta} + c \dot{\theta} + mgd\theta \quad (2.41)$$

Aplicando a transformada de Laplace para encontrar a função de transferência do subsistema, tem-se:

$$F_e(s) = s^2 J_b \theta(s) + sc\theta(s) + mgd\theta(s) \quad (2.42)$$

$$F_e(s) = (s^2 J_b + sc + mgd)\theta(s) \quad (2.43)$$

$$\frac{\theta(s)}{F_e(s)} = \frac{1}{s^2 J_b + sc + mgd} \quad (2.44)$$

Agora que foi obtido o modelo do braço do aeropêndulo, pode-se usar a relação linearizada $F_e = K_m \dot{\omega}$ para usar a saída do modelo do motor cc série como entrada do modelo do braço do aeropêndulo, como mostrado na equação 2.47.

$$\frac{\theta(s)}{K_m \dot{\omega}(s)} = \frac{1}{s^2 J_b + sc + mgd} \quad (2.45)$$

$$\frac{\theta(s)}{\dot{\omega}(s)} = \frac{K_m}{s^2 J_b + sc + mgd} \quad (2.46)$$

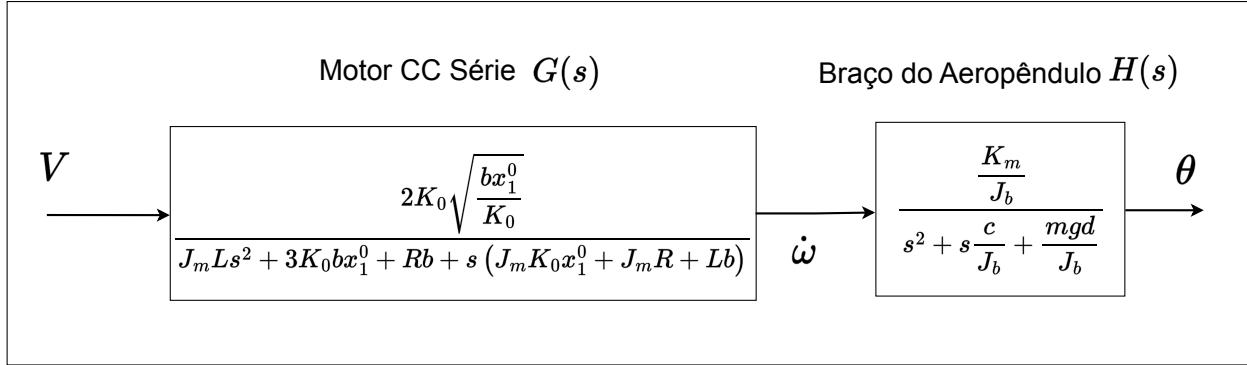
$$H(s) = \frac{\theta(s)}{\dot{\omega}(s)} = \frac{\frac{K_m}{J_b}}{s^2 + s \frac{c}{J_b} + \frac{mgd}{J_b}} \quad (2.47)$$

A equação 2.47 é a função de transferência do braço do aeropêndulo.

2.1.3 Junção dos subsistemas

A partir dos modelos encontrados nas subseções 2.1.1 e 2.1.2 pode-se obter o modelo completo do sistema unindo os subsistemas como mostra a figura 5. com isso, é possível modelar o Aeropêndulo tendo como entrada a tensão no motor CC série e como saída o ângulo do braço.

Figura 5 – Diagrama da junção dos subsistemas do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

Para obter uma função de transferência que relate a tensão dos terminais do motor com o ângulo do braço do Aeropêndulo, pode-se multiplicar as funções de transferências 2.39 e 2.47 dos subsistemas como mostrado em 2.48, já que estão em série.

$$\frac{\theta(s)}{V(s)} \cdot \frac{\dot{\omega}(s)}{\dot{\omega}(s)} = \frac{2K_0 \sqrt{\frac{bx_1^0}{K_0}}}{J_m L s^2 + 3K_0 b x_1^0 + Rb + s (J_m K_0 x_1^0 + J_m R + Lb)} \cdot \frac{\frac{K_m}{J_b}}{s^2 + s \frac{c}{J_b} + \frac{mgd}{J_b}} \quad (2.48)$$

Com isso, encontra-se um modelo matemático linear para representar o sistema, como mostrado em 2.49.

$$\frac{\theta(s)}{V(s)} = \frac{2K_0 \sqrt{\frac{bx_1^0}{K_0}} \cdot \frac{K_m}{J_b}}{J_m L s^2 + 3K_0 b x_1^0 + Rb + s (J_m K_0 x_1^0 + J_m R + Lb) \cdot \left(s^2 + s \frac{c}{J_b} + \frac{mgd}{J_b} \right)} \quad (2.49)$$

2.2 Implementação do Protótipo

2.2.1 Protótipo Aeropêndulo

O projeto para o desenvolvimento do protótipo consiste em uma estrutura mecânica e componentes eletrônicos, a Figura 6 mostra o protótipo finalizado com a parte estrutural e elétrica montada.

Figura 6 – Protótipo do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

2.2.2 Parte estrutural do sistema

O material escolhido para a estrutura foi o compensado, as chapas de compensado são materiais versáteis e amplamente utilizados na indústria da construção, marcenaria e artesanato. Elas são fabricadas a partir de finas lâminas de madeira, conhecidas como lâminas de folheado, coladas umas sobre as outras com fibras perpendiculares, conferindo maior estabilidade e resistência mecânica ao produto final, Figura 7.

Já para o braço do Aeropêndulo foi usando Tubo de Fibra de Carbono 3x3x2mm, Figura 8.

Figura 7 – Chapas de compensado.



Fonte: Autor.

Figura 8 – Tubo de Fibra de Vidro de Carbono 3x3x2mm.



Fonte: Autor.

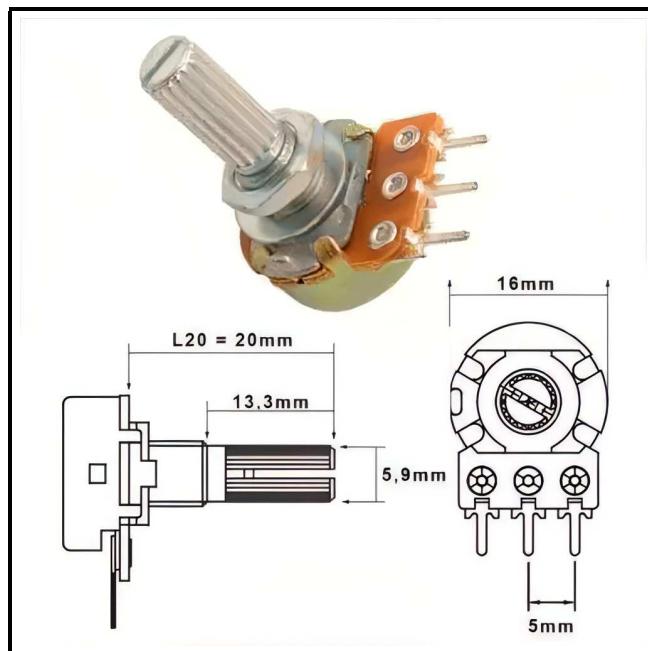
2.2.3 Parte Elétrica do sistema

Foram empregados os seguintes componentes eletrônicos no projeto: um Potenciômetro $50k\Omega$, uma Placa de desenvolvimento Esp32, um Módulo Driver L298n, um Conjunto (suporte/motor/hélice para drones fpv racing quadcopter) e componentes eletrônicos (resistor, capacitor).

2.2.3.1 Potenciômetro $50k\Omega$

O potenciômetro desempenha o papel de *encoder* ao obter o ângulo de inclinação do braço do Aeropêndulo. Essa funcionalidade é viabilizada pelo fato de que, conforme o braço se movimenta, a resistência do potenciômetro se altera, resultando em uma variação na tensão registrada em seus terminais. Essa interação permite estabelecer uma correlação entre a mudança na tensão elétrica e o ângulo de inclinação do braço do Aeropêndulo de maneira precisa e controlada.

Figura 9 – Potenciômetro $50k\Omega$.



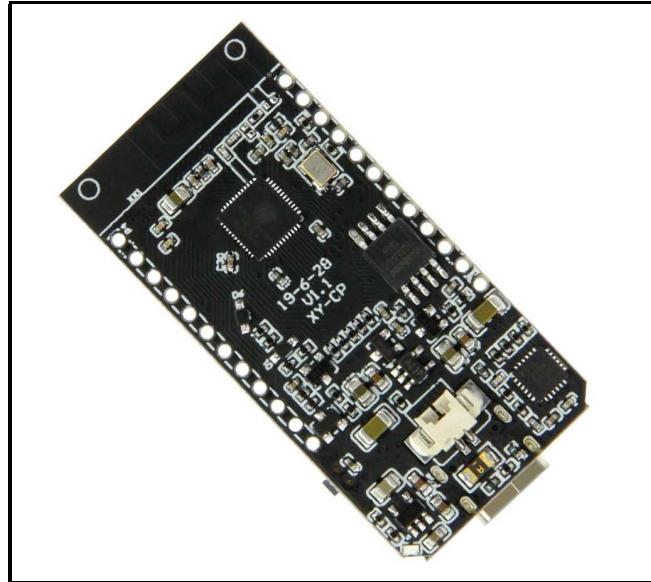
Fonte: Autor.

2.2.3.2 Placa de desenvolvimento Esp32

A placa de desenvolvimento ESP32 é uma poderosa e versátil plataforma que integra o popular microcontrolador ESP32 desenvolvido pela empresa *Espressif*. Projetada para atender às demandas de projetos de Internet das Coisas (IoT) e aplicações de conectividade sem fio, a ESP32 oferece uma vasta gama de recursos e funcionalidades. Equipada com um processador dual-core de 32 bits, Wi-Fi, Bluetooth, interfaces GPIO, I2C, SPI e UART, bem como suporte para memória externa.

A placa desempenhará uma função central no projeto, servindo tanto para a implementação do controlador discreto quanto para a obtenção de dados em tempo real relacionados aos estados do sistema. Esses estados incluem a leitura do sensor de ângulo do braço, a aquisição do sinal de controle e o cálculo do sinal de erro. Adicionalmente, a placa será empregada na geração dos sinais de referência necessários para o funcionamento ideal do sistema. A figura 10 ilustra a placa de desenvolvimento Esp32.

Figura 10 – Placa de desenvolvimento Esp32.



Fonte: Autor.

2.2.3.3 Fonte Chaveada 2A, 5V, 25W

Para a alimentação do motor CC série, foi usado uma fonte de alimentação de 5V/5A, Figura 11.

Figura 11 – Fonte Chaveada 2A, 5V, 25W.

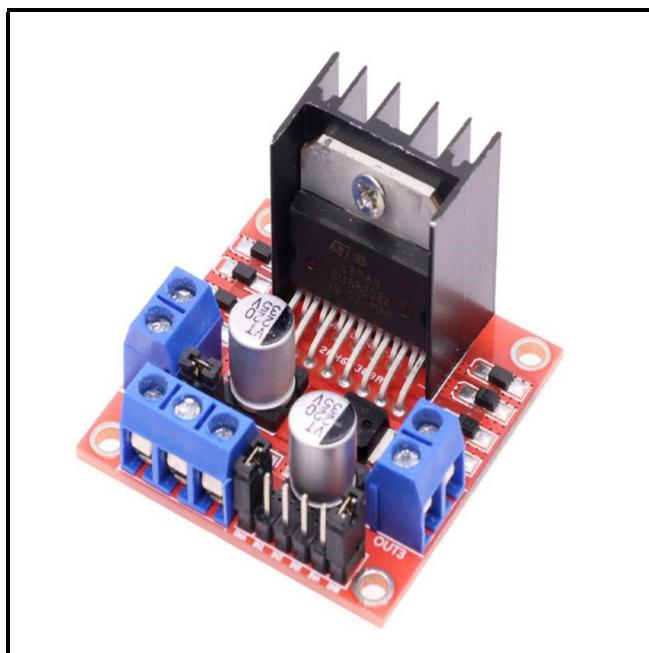


Fonte: Autor.

2.2.3.4 Módulo Driver L298n

Para regular a velocidade no eixo do Motor CC série, empregou-se um Módulo Driver L298n, o qual possibilita controlar a injeção de potência entregue ao motor mediante a aplicação de um sinal PWM em sua entrada. Dessa forma, ao ajustar o sinal PWM, é possível obter uma tensão controlada aplicada de maneira precisa aos terminais do motor. A Figura 12 ilustra o componente em questão.

Figura 12 – Módulo Driver L298n.



Fonte: Autor.

2.2.3.5 Conjunto (suporte motor cw/ccw hélice) para drones fpv racing quadcopter

Para obter-se a força de empuxo na extremidade do braço do Aeropêndulo foi usado um Conjunto (suporte/motor/hélice) para drones fpv racing quadcopter. A Figura 13 ilustra o componente em questão.

Figura 13 – Conjunto (suporte motor cw/ccw hélice).



Fonte: Autor.

2.2.3.6 Componentes eletrônicos (resistivos e capacitivos)

Por fim, para que o sinal do sensor (Potenciômetro) seja de boa qualidade, foi usado um filtro RC série, dessa forma, empregou-se um capacitor de um resistor na implementar do filtro. A Figura 14a corresponde aos componentes resistivos e a Figura 14b aos componentes capacitivos.

Figura 14 – Componentes eletrônicos.

(a) Resistores.



Fonte: Autor.

(b) Capacitores.



Fonte: Autor.

2.2.4 Montagem do Protótipo

2.2.4.1 Parte Física

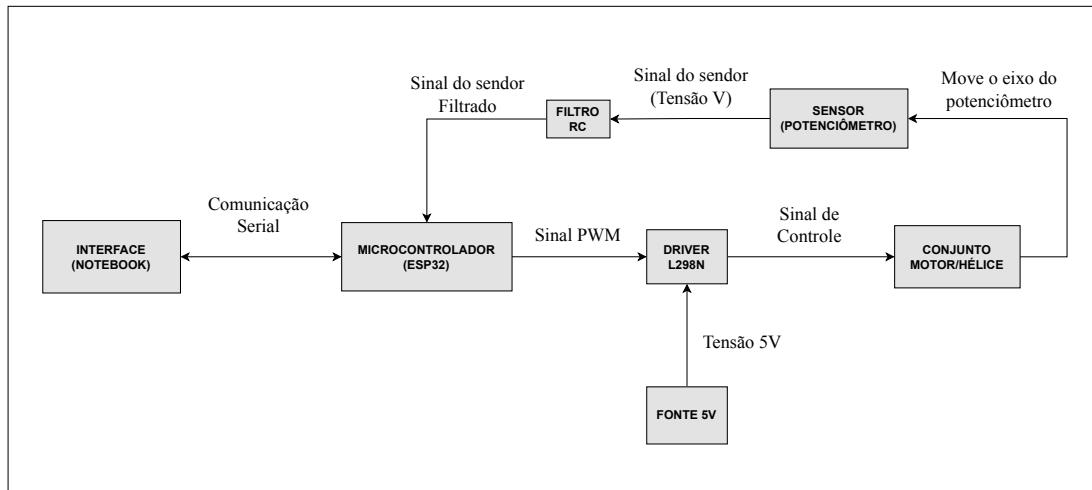
O protótipo foi concebido visando a desmontagem da estrutura, permitindo um transporte mais conveniente. A construção física compreende três componentes com pontos de conexão estratégicos. Ademais, o braço do aeropêndulo pode ser facilmente desacoplado no ponto de pivô, onde se conecta ao eixo do potenciômetro.

2.2.4.2 Parte Elétrica

A Figura 15 ilustra a dinâmica do fluxo de interligação do sistema elétrico do Aeropêndulo. Notavelmente, o microcontrolador desempenha um papel central ao gerar o sinal de controle PWM, realiza a leitura do sinal filtrado proveniente do sensor (Potenciômetro) e estabelecer comunicação com o computador através da interface serial. Adicionalmente, o driver é alimentado por uma fonte de tensão contínua de 5V, o qual amplifica e aplica o sinal de controle ao motor CC

Série. Esta ação, por sua vez, resulta em uma variação angular no braço do Aeropêndulo por conta do empuxo gerado pelas hélices, impactando diretamente o estado do sensor (Potenciômetro) e sua saída correspondente.

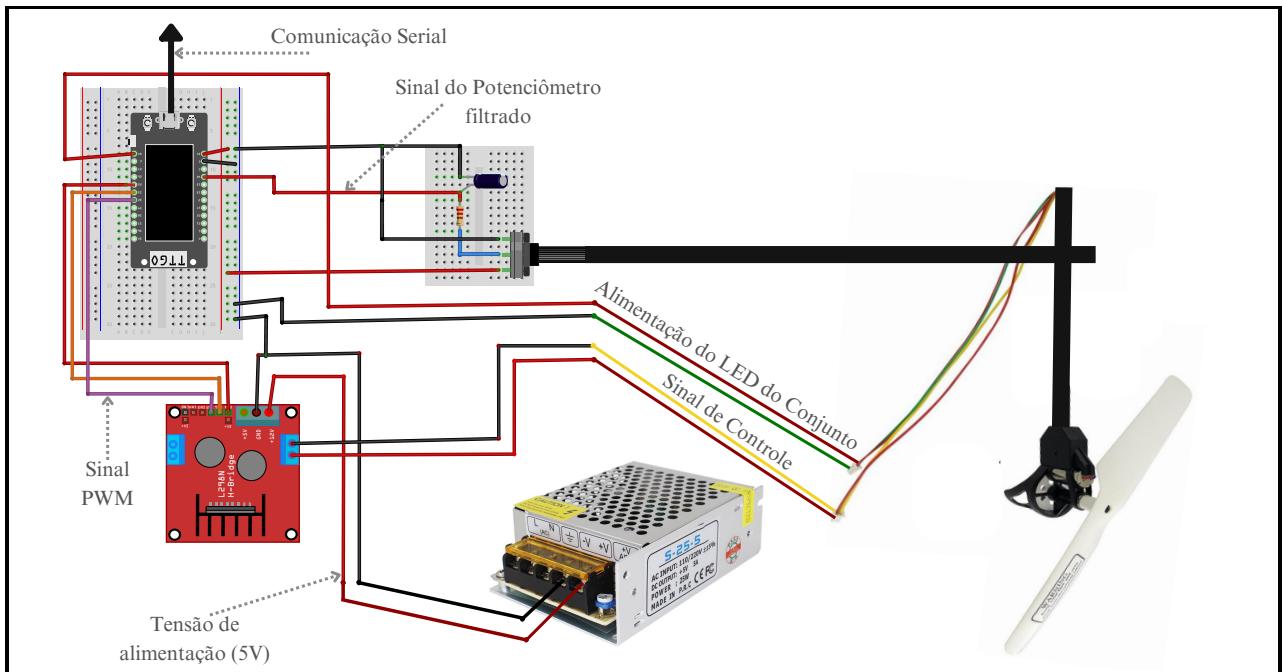
Figura 15 – Diagrama de comunicação do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

O esquema de ligação dos componentes do sistema elétrico é exemplificado na Figura 16.

Figura 16 – Esquema de conexões do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

2.3 Desenvolvimento dos Softwares

Para permitir a interação com o protótipo um conjunto de software foi desenvolvido, permitindo aos usuários aplicar diferentes conceitos de sistema de controle e obter uma visualização em tempo real dos estados do sistema através desses softwares, sendo eles, o firmware do microcontrolador, simulador 3D que usa o conceito de Gêmeo Digital explicado na subseção ?? e uma interface gráfica com um conjunto de funcionalidades capaz de manipular o sinal de referência e plotar os gráficos dos estados do sistema.

2.3.1 Linguagens Python, C e C++

Para o desenvolvimento dos softwares destinados à visualização de dados e ao simulador, a linguagem de programação escolhida foi o Python. Essa seleção se deu devido à natureza versátil da linguagem, que permite um desenvolvimento ágil de softwares para uma ampla gama de finalidades. Em paralelo, para a programação do microcontrolador, optou-se pelo uso das linguagens C/C++. Essa escolha se fundamenta na ampla adoção dessas linguagens na programação de sistemas embarcados, garantindo um ambiente propício para a eficaz implementação no microcontrolador.

A linguagem Python é uma linguagem de programação interpretada, de alto nível e de propósito geral. Ela é uma das linguagens de programação mais populares do mundo, sendo utilizada em uma ampla gama de aplicações, incluindo computação científica, ciência de dados, engenharia de software e inteligência artificial.

Python é uma linguagem de programação fácil de aprender e usar, mesmo para pessoas com pouca experiência em programação. Ela possui uma sintaxe clara e concisa, o que torna o código mais legível e manutenível. Além disso, Python é uma linguagem de programação poderosa e flexível, capaz de lidar com uma ampla gama de tarefas.

Por estas razões, Python é uma linguagem de programação ideal para a realização de trabalhos de pesquisa e desenvolvimento. Ela é uma ferramenta versátil e poderosa que pode ser utilizada para realizar uma ampla gama de tarefas, desde a análise de dados até a criação de aplicações complexas.

Já C e C++ são linguagens de programação que se concentram na eficiência e na portabilidade. Elas são amplamente utilizadas no desenvolvimento de sistemas operacionais, drivers de dispositivos, compiladores e outros softwares críticos.

C foi criada em 1972 por Dennis Ritchie para o desenvolvimento do sistema operacional Unix. C++ foi criada em 1983 por Bjarne Stroustrup como uma extensão de C para adicionar suporte à programação orientada a objetos.

C e C++ são linguagens de programação poderosas e flexíveis, mas também podem ser complexas e difíceis de aprender. Elas são recomendadas para desenvolvedores que precisam de

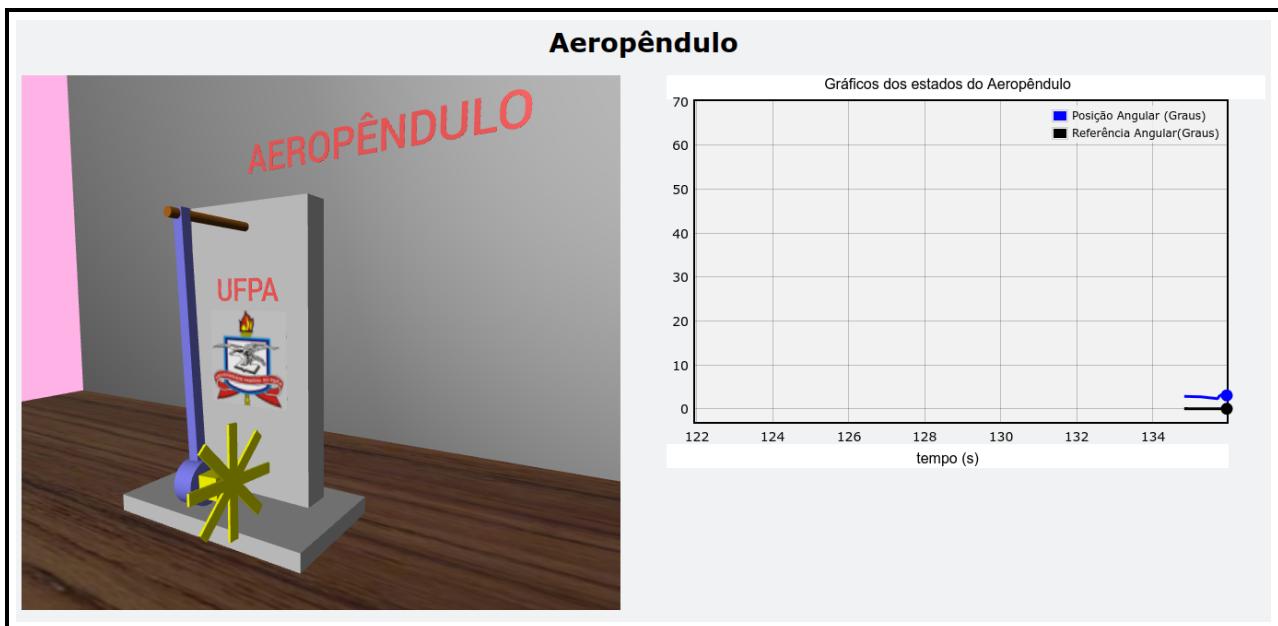
um alto nível de controle sobre o desempenho e a portabilidade de seu código.

2.3.2 Gêmeo Digital

Para desenvolver o simulador (Gêmeo Digital) foi usado a biblioteca Vpython, essa biblioteca possui um conjunto de funções que permite criar objetos 3D capazes de realizar movimentos rotacionais de translacionais, além disso, é possível plotar gráficos em tempo real. A Figura 18 ilustra a interface do simulador já finalizado, pode-se observar que a interface é composta de duas partes principais, uma que implementa o ambiente 3D com o Aeropêndulo e a outra gera os gráficos do sinal de referência e de saída.

Para atualizar os estados dos gráficos e a posição angular do gêmeo digital em tempo real a aplicação se comunica com a interface gráfica, subseção 2.3.3, com isso é possível realizar a atualização dos estados do gráfico assim como do gêmeo digital tendo como entrada os sinais dos estados do protótipo.

Figura 17 – Gêmeo Digital - Simulador com VPython.



Fonte: elaborado pelo autor (2023).

2.3.2.1 Biblioteca VPython

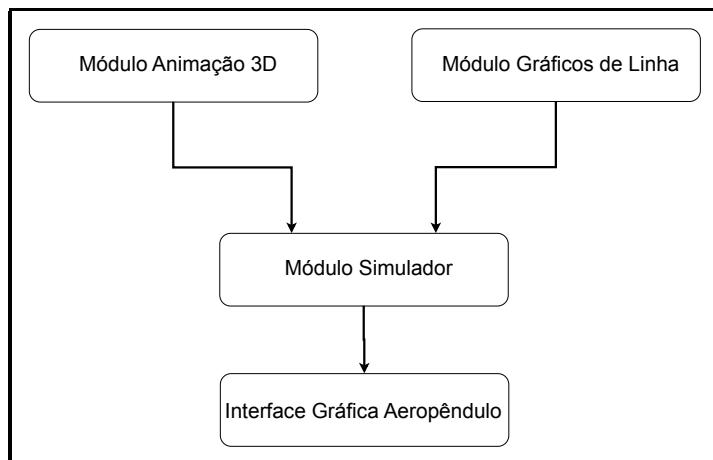
Conforme o site oficial (VPYTHON.ORG, 2023), "O VPython, por ser desenvolvido com a linguagem Python, é uma ferramenta versátil para a criação de animações 3D navegáveis. Ele é fácil de aprender e usar, mesmo para pessoas com pouca experiência em programação. No entanto, ele também oferece uma ampla gama de recursos para programadores e pesquisadores experientes."

Ao criar uma simulação com VPython e executá-la, o simulador será renderizado no navegador de internet padrão do sistema operacional.

2.3.2.2 Arquitetura do Gêmeo Digital

Para implementar o simulador a arquitetura do sistema consiste de um módulo para gerar os gráficos de linha e outro para desenhar o simulador 3D, existe um módulo que integra e atualiza os estados dos gráficos de linha e do simulador 3D, a Figura 18 mostra como a estrutura do simulador foi pensada, o último bloco é a interface gráfica, subseção 2.3.3, responsável por obter os estados do sistema real, com isso, é possível usar a velocidade angular real do protótipo como entrada para o gêmeo digital, isso faz com que a dinâmica do protótipo seja reproduzida no gêmeo digital, além disso, é possível obter os sinais de referência e de saída e atualizar os gráfios que compõem a interface do simulador.

Figura 18 – Diagrama da arquitetura do Gêmeo Digital.



Fonte: elaborado pelo autor (2023).

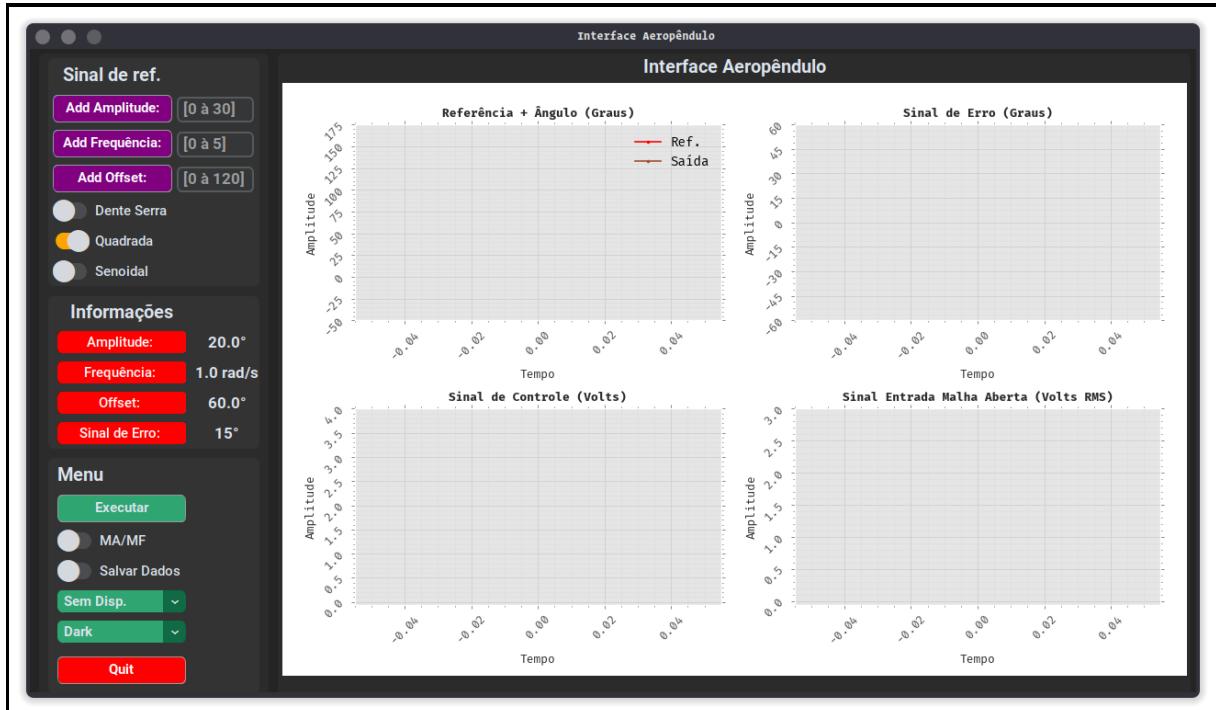
2.3.3 Interface Gráfica para Configuração, visualização e aquisição de dados da Planta

A interface gráfica consiste de um menu com diferentes opções de controle e visualização dos sinais do sistema e um conjunto de gráficos dinâmicos para visualizar os estados do sistema em tempo real. A Figura 19 mostra a interface do sistema. No entanto, além da parte visual existe a parte de aquisição e processamento de dados, em TI essa parte é chamada de Back-end, o projeto possui uma classe para realizar essa parte de aquisição e tratamento de dados, além de uma classe para detectar automaticamente o microcontrolador conectado a porta USB.

Para ser possível realizar os ensaios, modificação de parâmetros do sinal de referência e coletas de dados é preciso que o microcontrolador execute o firmware, subseção 2.3.4, desenvolvido

especificamente para esse propósito, dessa forma, a interface gráfica, Figura 19, conseguirá realizar o pre-processamento dos dados e plotar os sinais nos gráficos e demais atualizações nos softwares.

Figura 19 – Interface Gráfica.



Fonte: elaborado pelo autor (2023).

2.3.3.1 Biblioteca CustomTkinter e Matplotlib

Para o desenvolvimento da parte gráfica do software foi usado as bibliotecas CustomTkinter e Matplotlib, onde a biblioteca CustomTkinter implementa a estrutura de tela, botões, entradas de texto da aplicação e os módulos *Matplotlib.pyplot* e *Matplotlib.animation* possibilitam gerar gráficos em tempo real em conjunto com CustomTkinter.

Conforme ([SCKIMANSKY TOM, 2023](#)) explica no site oficial da biblioteca, "CustomTkinter é uma biblioteca de UI de desktop python baseada em Tkinter, que fornece widgets de aparência moderna e totalmente personalizáveis. Com CustomTkinter você terá uma aparência consistente em todas as plataformas de desktop (Windows, macOS, Linux)."

De acordo com ([MATPLOTLIB development team, 2023](#)), "Matplotlib é uma biblioteca abrangente para criar visualizações estáticas, animadas e interativas em Python. Matplotlib torna as coisas simples fáceis e as difíceis possíveis."

2.3.3.2 Biblioteca PySerial, NumPy e Pandas

Já para implementar o Back-end foram utilizadas as bibliotecas PySerial, NumPy e Pandas. Sendo que a biblioteca PySerial tem por finalidade realizar a comunicação entre o computador e o microcontrolador usando o protocolo serial via entrada USB, já o NumPy teve sua aplicação no processo de criação das matrizes contendo os dados lidos pela PySerial, por fim, para salvar os dados de ensaio foi usada a biblioteca Pandas, sendo salvos no formato CSV.

Conforme ([LIECHTI CHRIS, 2023](#)), "A biblioteca PySerial possibilita a comunicação serial entre o computador e o microcontrolador usando conexão USB, Este módulo encapsula o acesso à porta serial. Ele fornece backends para Python rodando em Windows, OSX, Linux, BSD (possivelmente qualquer sistema compatível com POSIX) e IronPython. O módulo denominado "serial" seleciona automaticamente o backend apropriado."

De acordo com ([NETO ANSELMO, 2023](#)), "O NumPy, uma biblioteca essencial para Python, oferece recursos robustos para realizar cálculos numéricos, sendo seu elemento central o ndarray, também conhecido como tensor. O ndarray se destaca por sua homogeneidade, exigindo que todos os elementos pertençam ao mesmo tipo, em contraste com as listas Python."

Segundo ([PANDAS development team , 2023](#)), site oficial da ferramenta, "pandas é uma ferramenta de análise e manipulação de dados de código aberto rápida, poderosa, flexível e fácil de usar, construído sobre a linguagem de programação Python."

2.3.3.3 Descrição das partes da interface gráfica

Nessa subseção será detalhado as partes do software, a Figura 20 está enumerando as partes e suas descrições estão logo abaixo.

Item 1: Sinal de Referência

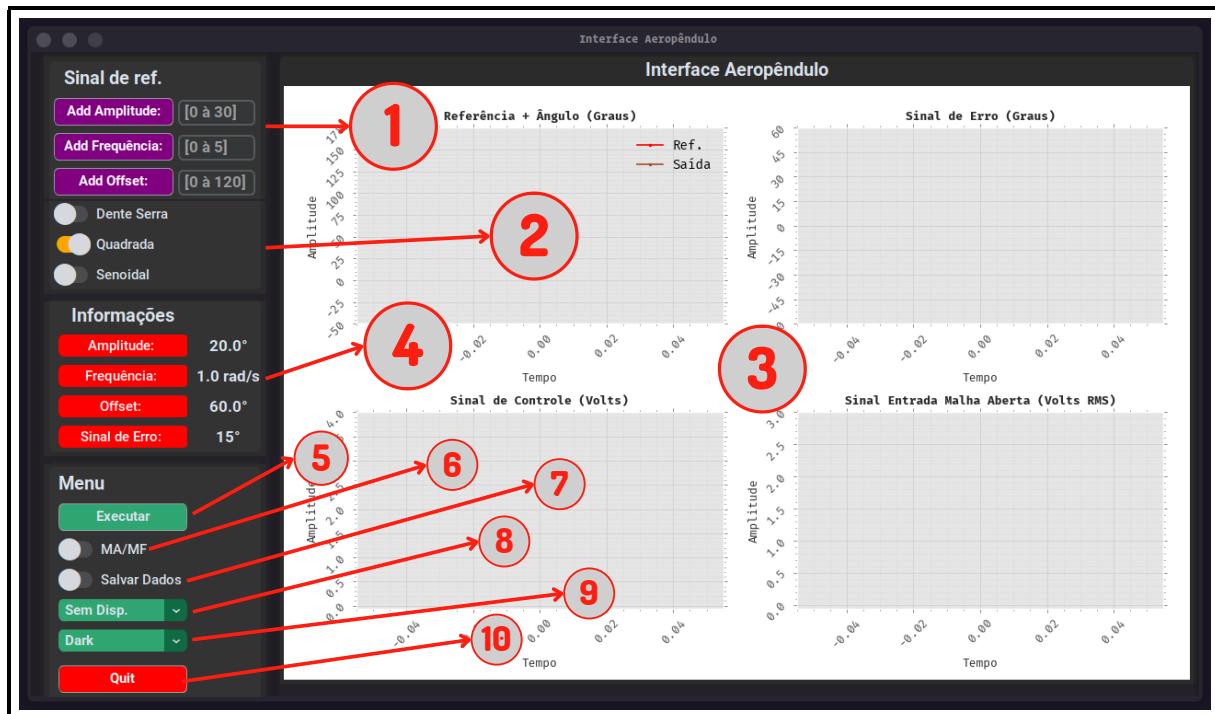
Essa parte do software configura os sinais de referência aplicado ao sistema em malha fechada.

- **Add Amplitude:** Configura a amplitude do sinal aplicado a entrada do sistema;
- **Add Frequência:** Configura a frequência do sinal aplicado a entrada do sistema;
- **Add Offset:** Configura a amplitude do offset, ponto de equilíbrio, aplicado a entrada do sistema.

Item 2: Seleção do Sinal de referência

- **Onda Serra:** Ao selecionar essa opção o sinal aplicado ao sistema será um dente de serra com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens;

Figura 20 – Partes da Interface Gráfica.



Fonte: elaborado pelo autor (2023).

- **Quadrada:** Ao selecionar essa opção o sinal aplicado ao sistema será uma onda quadrada com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens.;
- **Senoidal:** Ao selecionar essa opção o sinal aplicado ao sistema será uma onda senoidal com as configurações de amplitude, frequência e offset conforme configurado nos três primeiros sub itens.

Item 3: Gráficos

- **Gráficos:** Nesse item que são plotados os gráficos dos estados do sistema em tempo real.

Item 4: Informações

- **Amplitude:** Mostra a valor da amplitude do sinal de referência;
- **Frequência:** Mostra a valor da frequência do sinal de referência;
- **Offset:** Mostra a valor de offset aplicado ao aeropêndulo, esse valor é adicionado ao sinal de referência, dessa forma o sinal de referência varia em torno de um ponto de equilíbrio;
- **Sinal de Erro:** Mostra a valor do sinal de erro em tempo real.

Item 5: Menu

- **Executar:** Inicializa o ensaio, para isso é preciso que o microcontrolador esteja conectado ao computador.

Item 6: Menu

- **MA/MF:** Comuta entre o sistema em malha aberta e malha fechada.

Item 7: Menu

- **Salvar Dados:** Quando está ativado os dados são salvos em um arquivo CSV com data, hora, minuto e segundo do instante do salvamento, isso facilita que diferentes ensaios sejam realizados sem que os dados do anterior seja sobreescrito.

Item 8: Menu

- **Selecionar Dispositivo:** A aplicação detecta todos os microcontroladores conectados no computador e permite que o usuário realize a seleção do dispositivo desejado para realizar o ensaio.

Item 9: Menu

- **Selecionar Tema:** O software possui tema escuro e claro e essa opção permite que o usuário selecione entre essas duas opções.

Item 10: Menu

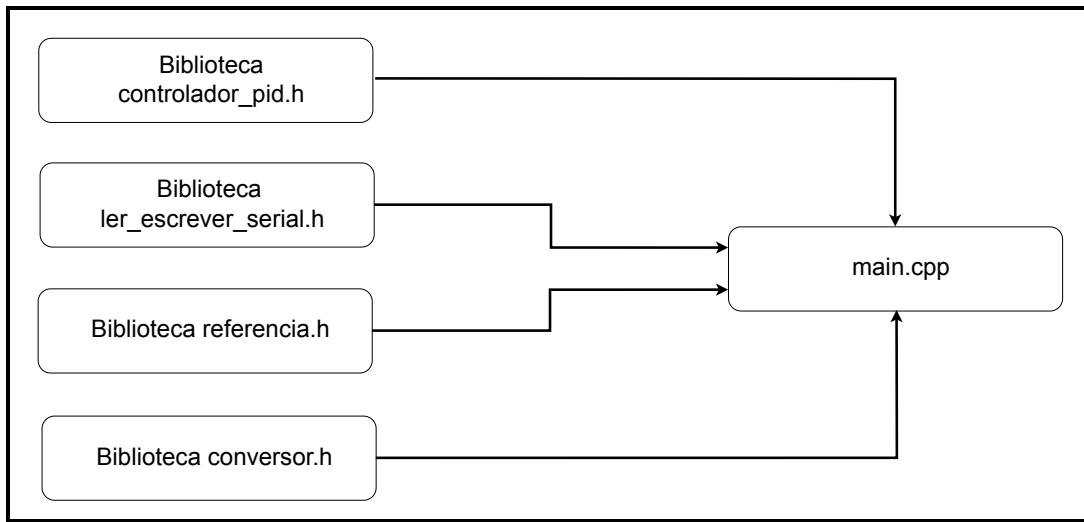
- **Quit:** Botão responsável por fechar a aplicação.

2.3.4 Firmware do microcontrolador

O firmwawe para o aeropêndulo foi desenvolvido a partir de bibliotecas, cada biblioteca tem sua funcionalidade, com isso o projeto se torna mais flexível, podendo ser atualizado e incrementado a medida que o projeto se torne mais robusto.

Para desenvolver o firmware, utilizamos a ferramenta de plataforma cruzada PlatformIO. Essa ferramenta foi criada com o propósito de centralizar o desenvolvimento de firmware, possibilitando a programação para diversos microcontroladores, como o ESP32, a Família do Arduino, STM32 e muitos outros. Isso torna o projeto ainda mais flexível e versátil. A arquitetura é descrita na Figura 21.

Figura 21 – Arquitetura do Firmware do Aeropêndulo.



Fonte: elaborado pelo autor (2023).

2.3.4.1 Biblioteca ler_escrever_serial

Para realizar a configuração dos parâmetros do sinal de referência em tempo real e o envio dos sinais do sistema via porta serial foi desenvolvido a biblioteca `ler_escrever_serial`, assim, a parte de recebimento e envio de dados do sistema fica centralizada permitindo sua melhoria e modificação sem interferir na lógica do código principal "`main.cpp`".

2.3.4.2 Biblioteca referencia

O sistema em malha fechada possui uma entrada de referência que o controlador tem por finalidade rastrear, para tornar o projeto do firmware mais legível criou-se o módulo `referencia.h` em que implementa inicialmente três sinais de referência com a possibilidade de configurar os parâmetros de Amplitude, frequência e offset, sendo esses sinais Onda Quadrada, Onda Senoidal e Onda Dente de Serra. Caso, observe-se a necessidade de outros sinais, basta adicionar a biblioteca.

2.3.4.3 Biblioteca conversor

Essa biblioteca possui uma classe `cpp` com métodos de conversões de diferentes grandezas, a primeira conversão é do sinal de um potenciômetro para ângulo, o segundo método converte o sinal de controle para ciclos PWM, o terceiro método converte de grau para radiano e o quarto de radiano para grau.

2.3.4.4 Biblioteca controlador_pid

Essa biblioteca implementa uma classe `cpp` para o controlador PID para ser empregado no protótipo quando selecionado a opção em malha fechada na interface gráfica.

2.3.4.5 Arquivo Principal mian.cpp

A ferramenta PlatformIO possui uma estrutura que permite a criação de bibliotecas e um arquivo main.cpp que implementa a lógica do algoritmo, com isso é possível incluir as bibliotecas desenvolvidas e criar o algoritmo para o que se deseja.

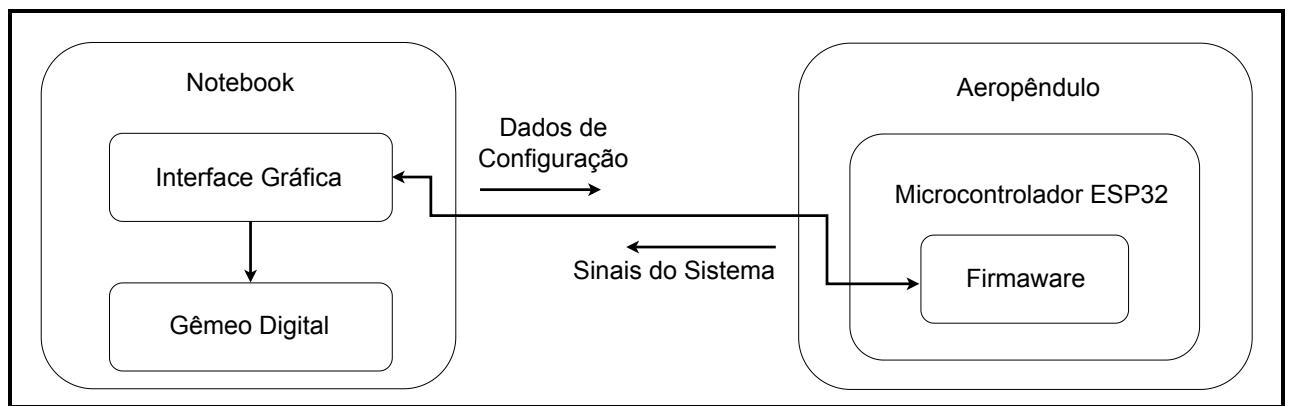
Ou seja, o firmware para o projeto possui várias funcionalidades que passam pela implementação do sinal de referência, leitura do sensor potenciômetro, envio e recebimento de dados via porta serial e implementação do controlador.

Por fim, com o firmware finalizado, o envio para o ESP32 é realizado usando o PlatfrmIO que concretiza a compilação e escrita no microcontrolador via porta serial.

2.4 Ecossistema do Projeto

Dado o caráter multifacetado deste projeto, a compreensão das interações entre seus diversos subsistemas pode apresentar desafios significativos. Com o propósito de simplificar e aprofundar esse processo de compreensão, a Figura 22 oferece uma representação visual por meio de um diagrama de blocos que esclarece as dinâmicas entre os distintos subsistemas desenvolvidos. Essa abordagem contribui para facilitar o entendimento holístico do sistema na totalidade, proporcionando um recurso valioso para pesquisadores futuros que desejam explorar e aprimorar este ecossistema.

Figura 22 – Diagrama do ecossistema do Projeto.



Fonte: elaborado pelo autor (2023).



RESULTADOS E DISCUSSÕES

3.1 Desenvolvimento do Protótipo e Softwares

Os sistemas propostos tanto físico quanto softwares foram desenvolvidos de forma exitosa, os procedimentos para a implementação se deu a partir de uma pesquisa bibliográfica e criativa, tendo como resultado o protótipo, Figura 6, e os softwares, Figuras 18, 18 e 21.

3.2 Identificação de sistema aplicado ao Aeropêndulo

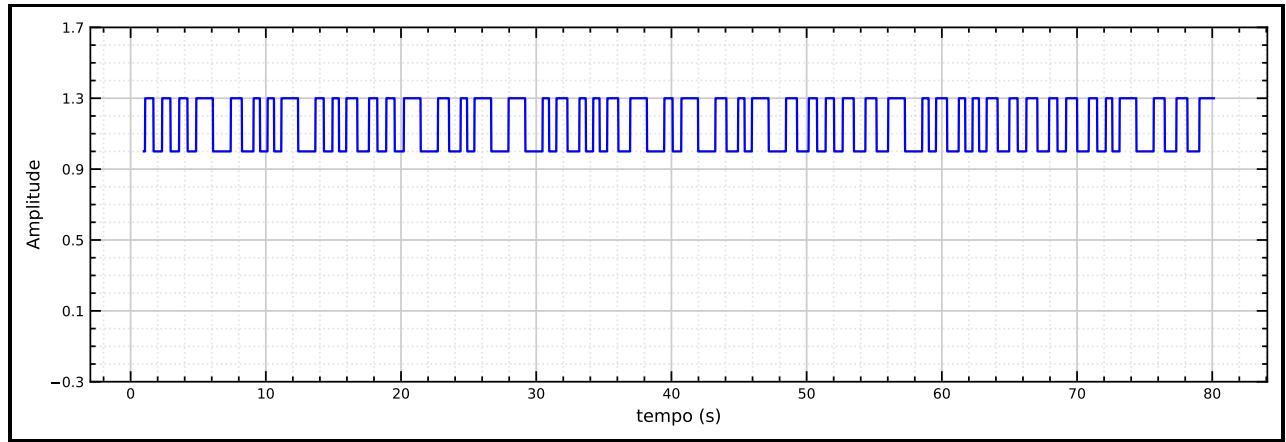
Com os subsistemas desenvolvidos, foram realizados alguns testes para validá-los, assim, o primeiro teste aplicado ao conjunto se tratou da identificação de sistemas. Após a identificação realizou-se uma simulação de modo a comparar o sinal de saída real com o simulado, visando visualizar a dinâmica do sistema identificado e poder comparar com o sistema real.

A partir do ecossistema desenvolvido para o projeto é possível usá-lo para estudar diferentes aspectos na área de sistemas de controle, uma delas é a identificação de sistemas, para esse trabalho o método em questão foi usado para avaliar o correto funcionamento projeto. Desse forma, o trabalho ([KLARISSA, 2023](#)) foi usando como base teórica para implementar o método.

3.2.0.1 Aplicação do Sinal PRBS na Entrada do Sistema

O sinal PRBS aplicado a entrada do sistema em malha aberta foi gerado no microcontrolador e convertido em sinal PWM para ser aplicado a entrada do Driver L298N, subseção 2.2.3.4, seus parâmetros foram definidos com uma amplitude de 0,3V, frequência fundamental de 0,4Hz e período de amostragem de 0,02 segundos, além disso, aplicou-se um offset de 1V (Ponto de operação), com isso, obteve-se o sinal PRBS de entrada mostrado na Figura 21.

Figura 23 – Sinal PRBS de entrada.

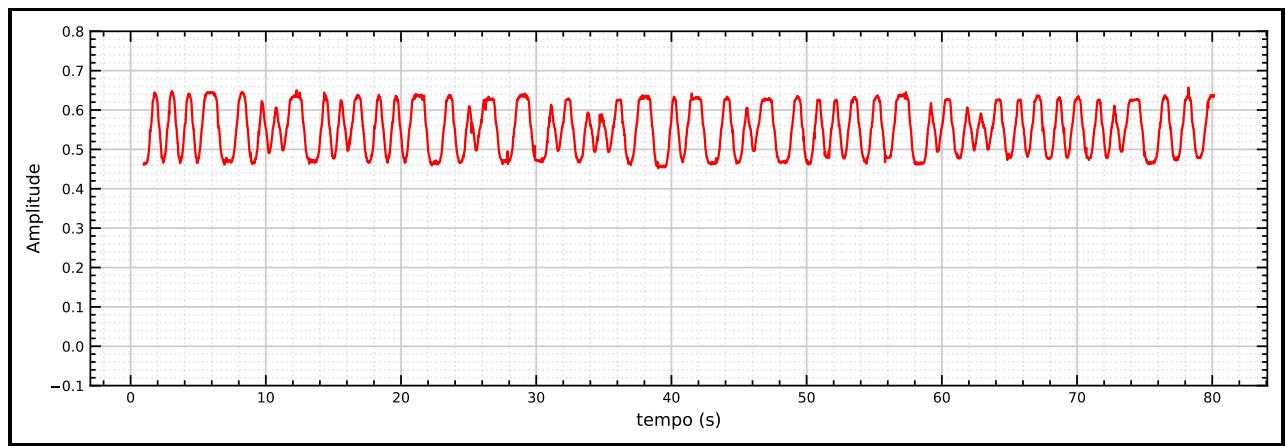


Fonte: elaborado pelo autor (2023).

3.2.0.2 Sinal de Saída Gerado a Partir do Sinal PRBS de Entrada

Para a obtenção do sinal de saída, que representa o ângulo do braço do aeropêndulo, foi empregado um método no qual um sinal PRBS foi aplicado à entrada do sistema. Para medir o ângulo, sinal de saída, utilizou-se um potenciômetro como sensor, cujas extremidades foram conectadas ao 3.3V e GND do ESP32. O terceiro fio do potenciômetro foi conectado a uma porta analógica do ESP32, permitindo, assim, a conversão da variação de tensão em valores angulares. A Figura 24 exibe o sinal de saída de um ensaio.

Figura 24 – Sinal PRBS de saída.



Fonte: elaborado pelo autor (2023).

3.2.1 Identificação Usando Mínimos Quadrados

Para realizar a identificação de sistemas deve-se retirar os offsets tanto do sinal de entrada quanto para o de saída, com essa etapa realizada, deve-se separar dois conjuntos de dados para

identificação e validação do modelo proposto, tanto para o sinal de entrada quanto para o de saída. Foi usando 60% dos dados para identificação de 40% para validação.

3.2.1.1 Passos para realização da identificação

Passo 1: Obter dados Entrada/Saída do sistema;

Passo 2: Dividir dados entre identificação e validação;

Passo 3: Definir a Função de Transferência;

Passo 4: Definir a matriz de regressão;

Passo 5: Encontrar o vetor de coeficientes pela formulação de mínimos quadrados;

Passo 6: Substituir os coeficientes na função de transferência escolhida anteriormente;

Passo 7: Validar o modelo do sistema a partir de dados reais.

3.2.1.2 Identificação Usando Função de Transferência Discreta de Segunda, Quinta e Décima Ordem

Para realizar o cálculo numérico usando a base de dados do ensaio foi usando a linguagem de programação Python, que possui um conjunto de bibliotecas que facilita o implementação dos cálculos. Após obter os dados de ensaio, os passos seguintes foram desenvolvidos em um Script Python, sendo o primeiro passo a importação das bibliotecas, como mostra a célula de código abaixo.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import control as ct
```

No segundo passo foi feito o carregamento dos dados de ensaio PRBS e realizada a separação entre dados de identificação e validação, para isso foi usado a biblioteca Pandas em conjunto com a biblioteca NumPy que possibilita carregar dados de diferentes formatos e manipulá-los, os dados de ensaio foram salvos em formato CSV (Comma-separated values), o código para realizar o carregamento dos dados e a separação é mostrado na célula de código abaixo.

```
file = "caminho completo dos dados/arquivo_9_9_2023_13_33_24.csv"
dados_malha_aberta_prbs = pd.read_csv(file, header=None, sep=',').values
dados_malha_aberta_prbs[0][0] = 0.0
dados_malha_aberta_prbs

# Variaveis contendo os dados de tempo, entrada e saida
tempo = np.array(dados_malha_aberta_prbs[:,7])
```

```

sinal_prbs_entrada = np.array(dados_malha_aberta_prbs[:, 6])
sinal_saida = np.array(dados_malha_aberta_prbs[:, 2])

# Convertendo o sinal de Graus para Radianos
sinal_saida = np.squeeze(np.deg2rad(sinal_saida))
tempo = tempo - min(tempo)

```

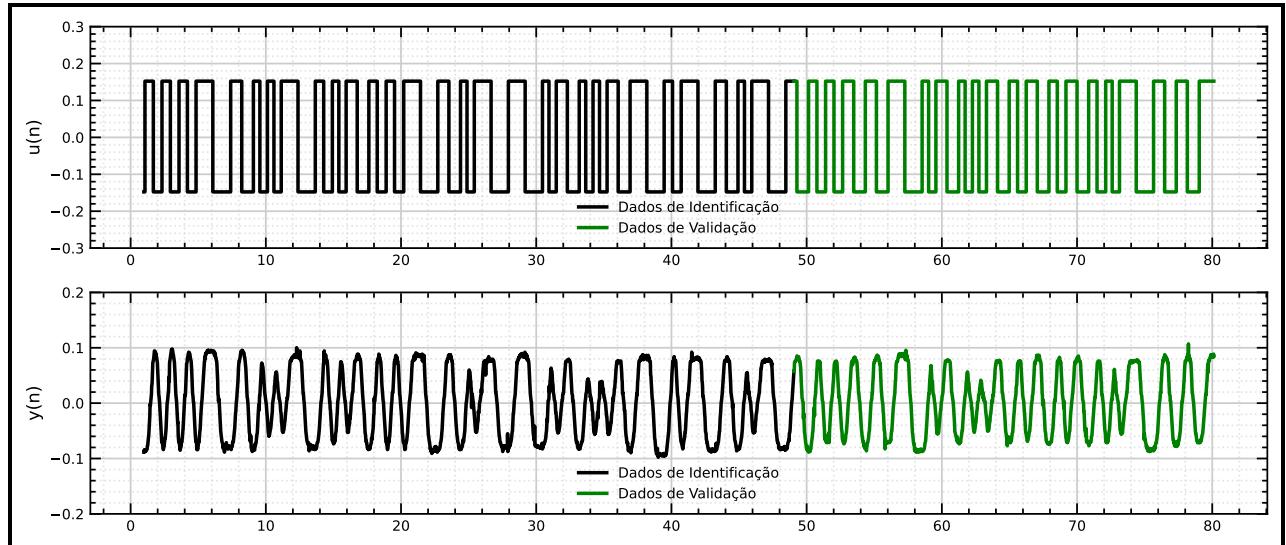
Além disso, é necessário remover as primeiras amostras por conter o transitório do sistema e retirar os offsets dos sinais de entrada e saída, o que é feito na célula de código abaixo. A figura 25 exibe os gráficos dos sinais de entrada/saída, podemos observar que os sinais estão centralizados em zero e não possui transitório no início do sinal.

```

u1 = sinal_prbs_entrada[50:] - np.mean(sinal_prbs_entrada[50:])
yout = sinal_saida[50:] - np.mean((sinal_saida[50:]))
t = tempo[50:]

```

Figura 25 – Dados de identificação e validação do modelo.



Fonte: elaborado pelo autor (2023).

Com os dados de ensaio pré-processados e prontos para aplicar a técnica de identificação, foi definida uma função de transferência discreta, essa função tem a finalidade de aproximar um modelo que satisfaça a dinâmica do aeropêndulo. Para um primeiro teste foi usada uma função de segunda ordem, como mostra a equação 3.1.

$$H(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (3.1)$$

Para aplicar o método dos mínimos quadrados e obter os coeficientes da função de transferência de segunda ordem precisa-se ter as matrizes de regressão tanto para os dados de entrada quanto para os de saída, o código abaixo cria o algoritmo necessário para se obter as matrizes a partir dos dados do ensaio.

```
# Matriz de regressao:
nb = 3; na = 2
ni = np.arange(na, Ni + na)
M = np.zeros((Ni, na + nb))

# Para regressores de y:
for l in np.arange(0, na):
    M[:, l] = yout[ni - l - 1]

# Para regressores de u:
for l in np.arange(0, nb):
    M[:, na + l] = u1[ni - l]

print(M.shape)
```

Com as matrizes de regressores definidas, foi realizada a aplicação do método dos mínimos quadrados, o que é exemplificado na célula de código abaixo, a biblioteca NumPy possibilita cálculos de álgebra matricial de forma rápida e simples.

```
# Minimos quadrados
thetaA = np.linalg.inv(M.T @ M) @ M.T @ yout[ni]
thetaA
```

Para a função de transferência de segunda ordem escolhida, equação 3.1, em que possui dois coeficientes no numerador e 3 no denominador, dessa forma, os coeficientes obtidos ao aplicar o método de mínimos quadrados retorna em suas duas primeiras posições os coeficientes do numerador e o restante são os coeficientes do denominador, a partir dos coeficientes obtidos, pode-se definir a função de transferência discreta do sistema, isso é feito no Python usando a biblioteca Python-Control que permite realizar ensaios simulados a partir da função de transferência.

```
a1, a2 = thetaA[:2]
b0, b1, b2 = thetaA[2:]

Ba = [b0, b1, b2]
Aa = [1, -a1, -a2]

Gz = ct.tf(Ba, Aa, Ts)
```

Função de transferência discreta, 3.2, com os coeficientes obtidos a partir do método de mínimos quadrados.

$$\frac{-0.002602z^2 + 0.004962z + 0.0163}{z^2 - 1.176z + 0.1849} \quad dt = 0.019 \quad (3.2)$$

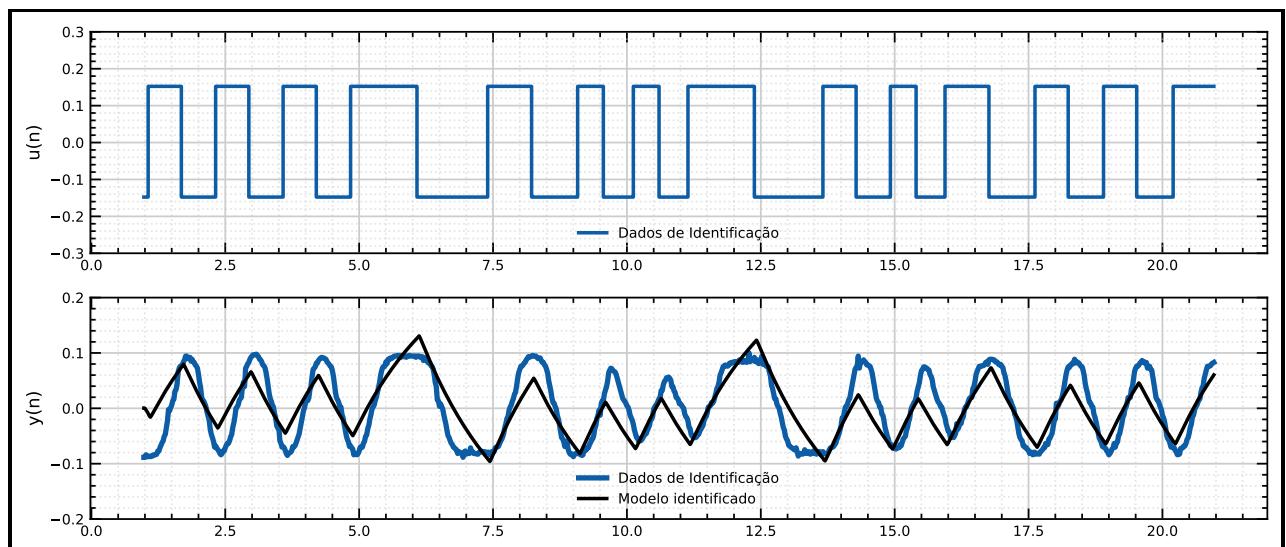
Para validar o modelo encontrado é preciso testar a função de transferência discreta usando os dados de entrada do ensaio real, com isso obtém-se os dados de saída da simulação, esse dado pode ser comparado com a saída do sistema real, caso ambos os gráficos aproximem suas dinâmicas pode-se concluir que o modelo corresponde com o sistema real, caso contrário é preciso definir uma nova função de transferência e realizar novamente o teste de validação.

A biblioteca Python-control possibilita simular a dinâmica de um sistema a partir da sua função de transferência discreta, podendo receber como argumento um vetor de entradas, como mostrado a célula de código abaixo.

```
_ , yp = ct.forced_response(Gz, U=u1)
```

Ao plotar os gráficos de saída real e simulada tem-se como resultado a Figura 26, é possível observar que a saída da simulação não corresponde de forma aceitável com a dinâmica do sinal de saída do sistema real, ou seja, o modelo não foi robusto o suficiente para identificar o sistema em questão.

Figura 26 – Validação do modelo de segundo grau.



Fonte: elaborado pelo autor (2023).

A partir de uma análise qualitativa, pode-se supor que o grau da função de transferência discreta escolhida não consegue aproximar a dinâmica real do aeropêndulo, dessa forma, foi

realizado alguns testes e identificado que uma função de transferência discreta de décima ordem, equação 3.3, aproxima de forma aceitável o sistema real.

$$H(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} + a_5 z^{-5} + a_6 z^{-6} + a_7 z^{-7} + a_8 z^{-8} + a_9 z^{-9} + a_{10} z^{-10}} \quad (3.3)$$

Assim, foi ajustado o algorítimo para identificar o sistema a partir da estrutura da equação 3.3, o código abaixo gera as matrizes tanto para os regressores de entrada quanto para o de saída.

```
# Matriz de regressao:
nb = 4
na = 10
ni = np.arange(na, Ni + na)
M = np.zeros((Ni, na + nb))

# Para regressores de y:
for l in np.arange(0, na):
    M[:, l] = yout[ni - l - 1]

# Para regressores de u:
for l in np.arange(0, nb):
    M[:, na+l] = u1[ni-l]

print(M.shape)
```

Com as matrizes de regressores ajustada para um sistema de décima ordem, foi realizada a aplicação do método dos mínimos quadrados.

```
# Minimos quadrados
thetaA = np.linalg.inv(M.T @ M) @ M.T @ yout[ni]
thetaA
```

A equação 3.3 possui 4 coeficientes no numerador e 10 no denominador, dessa forma, as quatro primeiras posições da variável **thetaA** corresponde aos coeficientes do numerador e o restante aos coeficientes do denominador, a célula de código abaixo define a função de transferência usando a estrutura da equação 3.3 e os coeficientes obtidos.

```
a1, a2, a3, a4, a5, a6, a7, a8, a9, a10 = thetaA[:10]
b0, b1, b2, b3 = thetaA[10:]

Ba = [b0, b1, b2, b3]
```

```
Aa = [1, -a1, -a2, -a3, -a4, -a5, -a6, -a7, -a8, -a9, -a10]

Gz = ct.tf(Ba, Aa, Ts)
```

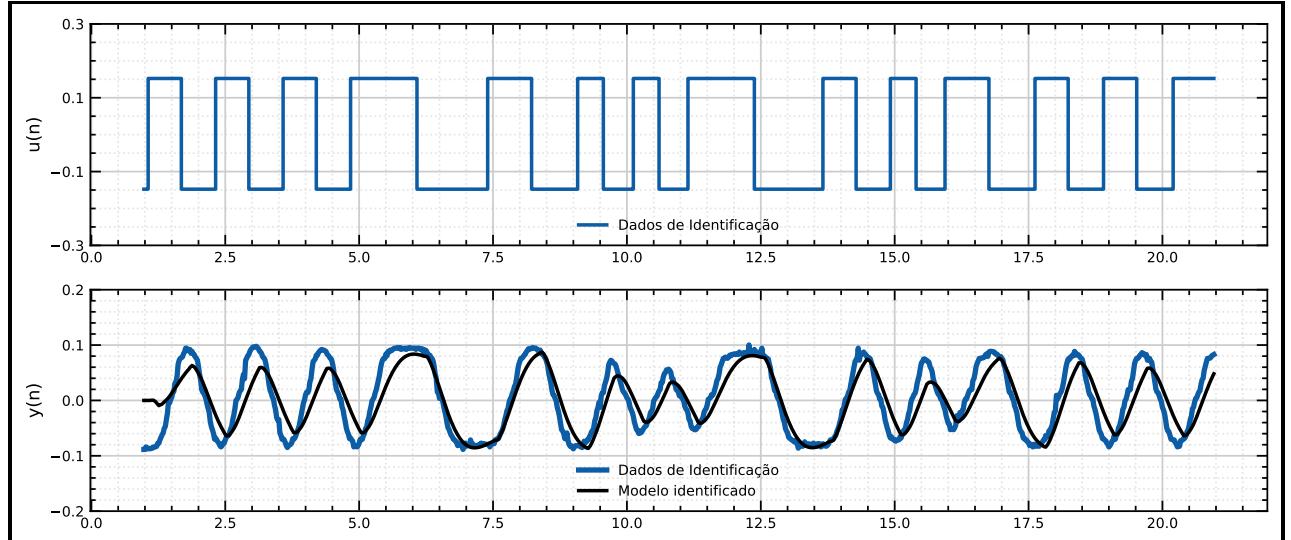
Assim, obtém-se a função de transferência discreta, equação 3.4, com os valores encontrado a partir do método de mínimos quadrados.

$$\frac{-0.0029z^3 + 0.0023z^2 + 0.0016z + 0.0097}{z^{10} - 0.9z^9 - 0.3z^8 - 0.014z^7 + 0.13z^6 + 0.15z^5 + 0.012z^4 - 0.05z^3 - 0.12z^2 - 0.02z + 0.15} \quad dt = 0.02 \quad (3.4)$$

Por fim, realizou-se novamente a simulação do modelo encontrado, com o propósito de compará-lo aos dados de saída obtidos durante o ensaio do sistema real, a célula de código abaixo implementa a simulação.

```
_ , yp = ct.forced_response(Gz, U=u1)
```

Figura 27 – Validação do modelo de segundo grau.



Fonte: elaborado pelo autor (2023).

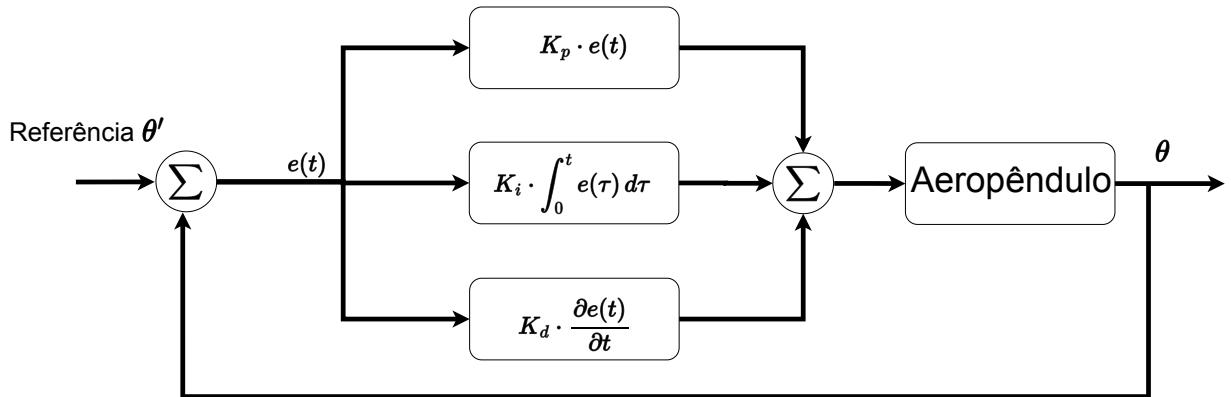
A partir de uma análise qualitativa entre o sinal de saída real e o simulado, figura 27, pode-se concluir que o modelo proposto a partir do método de mínimos quadrados teve sucesso, no entanto, ao modelar sistemas reais sempre haverá erro, isso por conta da não linearidade dos componentes físicos: sensores, atuadores e do próprio sistema, sendo assim, o modelo obtido é apenas uma aproximação do sistema real.

3.3 Ensaio em Malha Fechada com Controlador PID

O ecossistema desenvolvido possibilita aplicar diferentes controladores, bastando implementá-lo no formato de uma biblioteca e chamá-la no código principal do firmware, passando como parâmetro o erro que é calculado subtraindo da referência o valor lido na saída do sistema, isso permite maior flexibilidade ao pesquisador, que pode projetar diferentes controladores e realizar seus testes sem que seja necessário sobrescrever o código de implementação do sistema de controle, ou seja, a equação do controlador.

Com propósito de testar o sistema em malha fechada, foi desenvolvido um controlador PID simples, tendo apenas um requisito, erro nulo para o sinal degrau. A Figura 28 exibe o diagrama do controlador proposto.

Figura 28 – Sistema em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).

No contexto da validação do sistema em malha fechada, optou-se por uma abordagem simplificada para a sintonia do controlador PID. Este método baseou-se na técnica de tentativa e erro. Após uma série de experimentos, determinaram-se os valores ótimos para os coeficientes do controlador, que foram os seguintes: $K_p = 0,02$, $K_i = 0,055$ e $K_d = 0,35$.

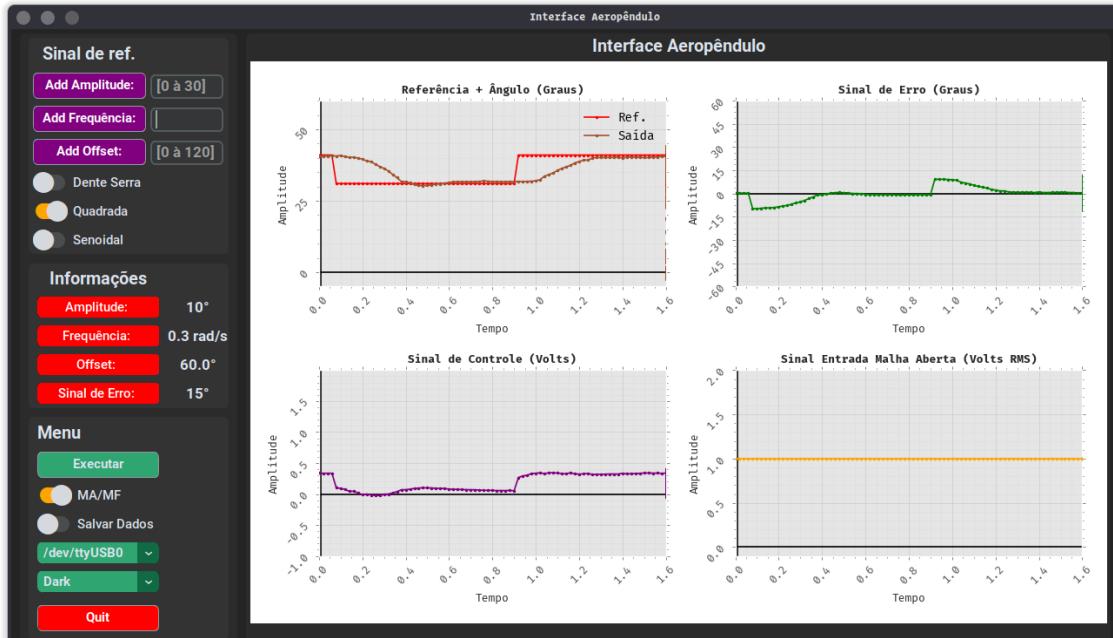
3.3.1 Onda Quadrada

Ao inicializar a interface é possível escolher a opção de malha fechada, com isso a interface envia um comando para o microcontrolador que fecha a malha e aplica o sinal de controle na entrada do sistema, dessa forma, o aeropêndulo rasteia o sinal de referência escolhido, para o primeiro teste foi escolhido como sinal de referência uma onda quadrada com os seguintes parâmetros, frequência de 0,5 Hz, Amplitude 15° e offset de 1V. A figura 29, permite visualizar os sinais de referência e saída, ao analisá-los pode-se observar que o sinal de saída rasteia o sinal de referência. Além disso, observa-se que o sistema possui um transitório nas extremidades do sinal de onda

quadrada, isso pode ser analisado a partir do sinal de erro que aumenta as extremidades do sinal de referência.

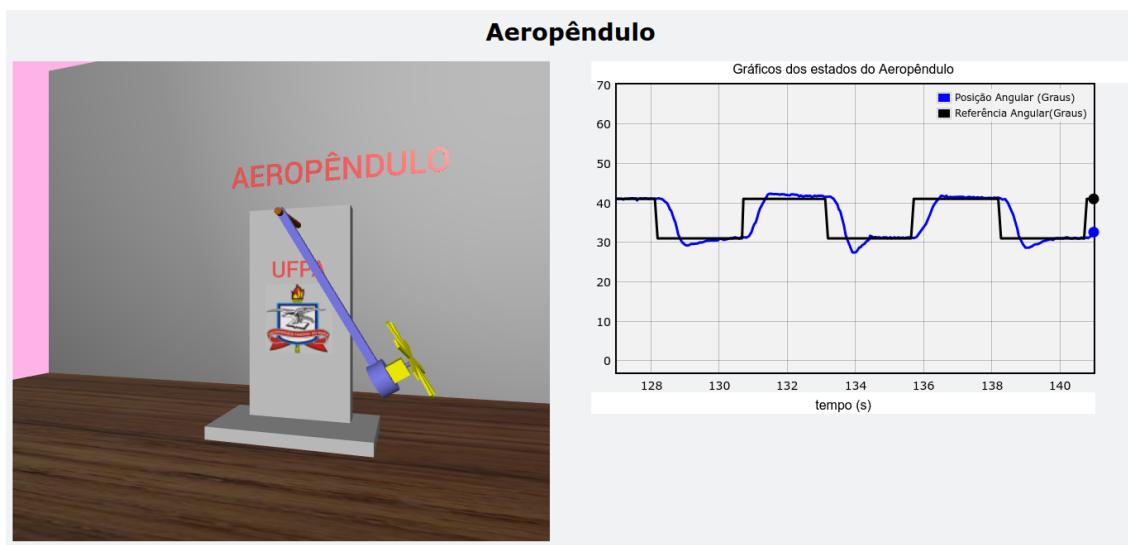
O gêmeo digital consome em tempo real o sinal angular do sistema real capturado através do potenciômetro, com isso é possível visualizar de uma forma gráfica a dinâmica do sistema real.

Figura 29 – Gráficos dos Estados do Aeropêndulo com Controlador PID.



Fonte: elaborado pelo autor (2023).

Figura 30 – Gêmeo Digital com Controlador PID.

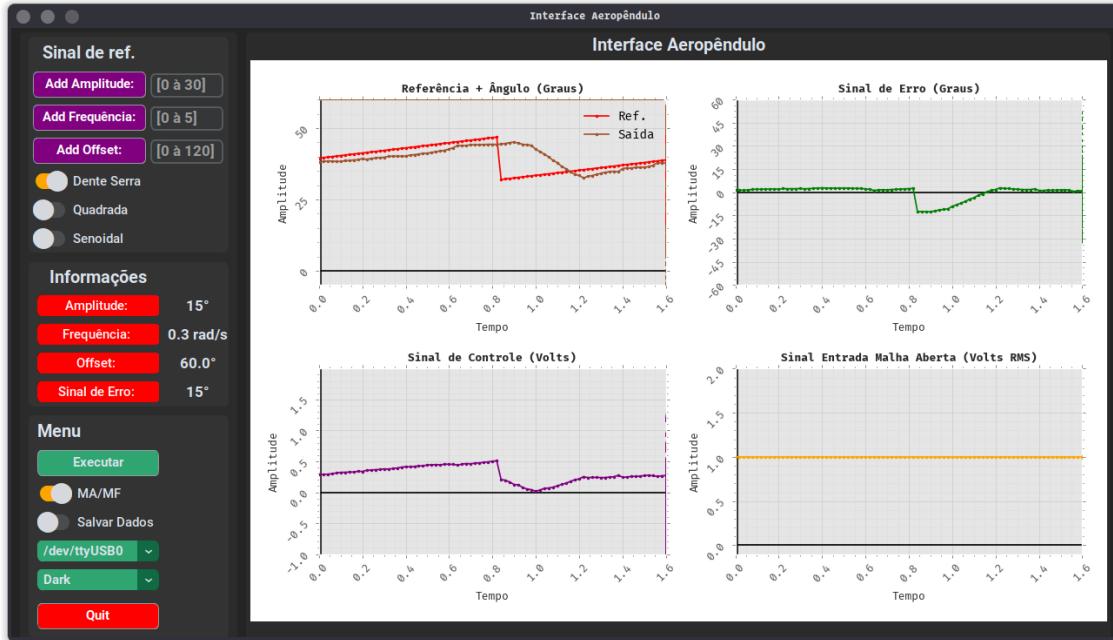


Fonte: elaborado pelo autor (2023).

3.3.2 Onda Dente de Serra

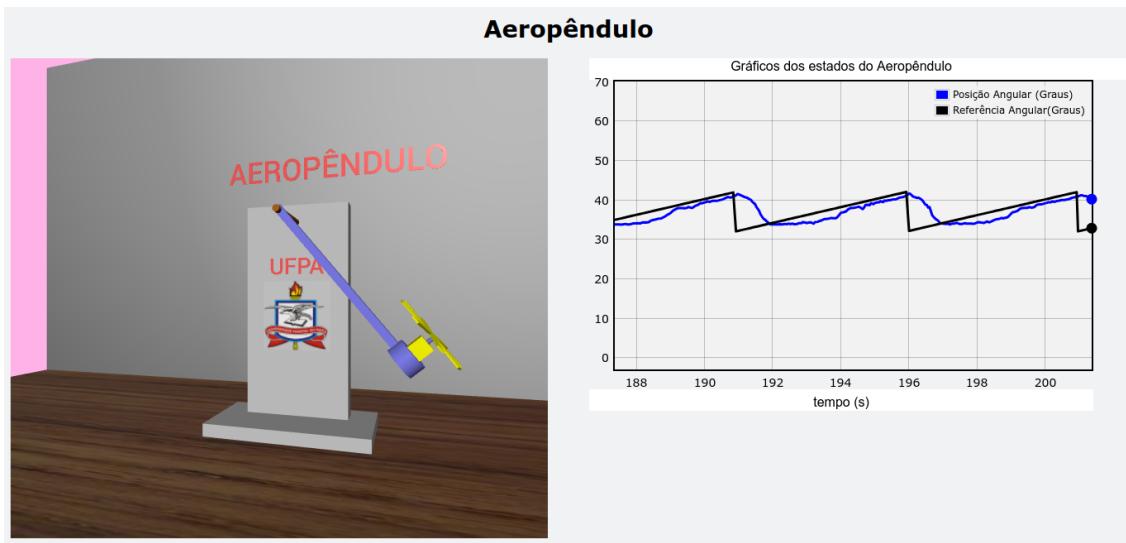
Com o sinal de referência sendo uma onda dente de serra, pode-se observar que o controlador consegue rastreá-lo, tendo assim como no sinal degrau, um transitório nas extremidades.

Figura 31 – Sistema em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).

Figura 32 – Sistema em Malha Fechada com Controlador PID.



Fonte: elaborado pelo autor (2023).



CONCLUSÃO

4.1 Considerações Finais

O ecossistema concebido para este projeto foi meticulosamente desenvolvido e testado, com resultados positivos abrangendo a modelagem matemática do sistema, a concepção e construção do protótipo, a elaboração do firmware, bem como o desenvolvimento dos softwares para a interface com o Aeropêndulo e a criação do gêmeo digital.

A modelagem matemática do sistema foi fundamentada nos princípios de Newton, no entanto, este processo demonstrou que a modelagem de sistemas pode rapidamente se tornar complexa e impraticável. Mesmo após a obtenção de um modelo que descreva a dinâmica do sistema, a determinação dos coeficientes finais pode ser uma tarefa árdua, uma vez que requer a utilização de sensores para obter tais grandezas, acrescentando uma camada de complexidade e desafios ao processo de modelagem. Dessa forma, o método de identificação de sistemas pode ser aplicado para encontrar um modelo matemático que aproxime a dinâmica do sistema real.

A concepção do protótipo foi cuidadosamente planejada com o intuito de simplificar o processo de construção. Para alcançar esse objetivo, optou-se pelo uso de materiais de baixo custo na estrutura e componentes eletrônicos acessíveis. Isso resultou na capacidade de reproduzir o sistema de maneira eficiente e acessível, tornando-o acessível não apenas a acadêmicos, mas também a entusiastas interessados em implementar o sistema.

4.2 Trabalhos Futuros

REFERÊNCIAS

- KLARISSA, S. Identificação não-linear no espaço de estados por regressão esparsa de bancada motor-gerador. 2023. 31
- LICEAGA-CASTRO, J. U.; SILLER-ALCALÁ, I. I.; JAIMES-PONCE, J.; ALCÁNTARA-RAMÍREZ, R. Series dc motor modeling and identification. In: *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. [S.l.: s.n.], 2017. p. 248–253. 5, 7, 8
- LIECHTI CHRIS. *pySerial documentação Oficial*. 2023. Disponível em: <<https://pyserial.readthedocs.io/en/latest/pyserial.html>>. Acesso em: 01 de outubro 2023. 26
- MATPLOTLIB development team. *Site oficial da biblioteca Matplotlib*. 2023. Disponível em: <<https://matplotlib.org>>. Acesso em: 01 de outubro 2023. 25
- MOHAMMADBAGHERI, A.; YAGHOOBI, M. A new approach to control a driven pendulum with pid method. In: *2011 UkSim 13th International Conference on Computer Modelling and Simulation*. [S.l.: s.n.], 2011. p. 207–211. 12
- NETO ANSELMO. *Introdução ao Numerical Python (Numpy)*. 2023. Disponível em: <<http://www.opl.ufc.br/post/numpy>>. Acesso em: 01 de outubro 2023. 26
- OGATA, K. *Engenharia de Controle Moderno*. Brasil: Pearson Education do Brasil, 2014. 4
- PANDAS development team . *Site oficial da biblioteca Pandas*. 2023. Disponível em: <<https://pandas.pydata.org>>. Acesso em: 01 de outubro 2023. 26
- SCKIMANSKY TOM. *Site oficial da biblioteca CustomTkinter*. 2023. Disponível em: <<https://customtkinter.tomschimansky.com>>. Acesso em: 01 de outubro 2023. 25
- UMANS, S. *Máquinas Elétricas de Fitzgerald e Kingsley - 7.ed.* AMGH Editora, 2014. ISBN 9788580553741. Disponível em: <<https://books.google.com.br/books?id=3Fa2AwAAQBAJ>>. 5
- VPYTHON.ORG. *Site oficial da biblioteca VPython*. 2023. Disponível em: <<https://customtkinter.tomschimansky.com>>. Acesso em: 01 de outubro 2023. 23